CALIFORNIA POLYTECHNIC STATE UNIVERSITY
WIND RESOURCE ASSESSMENT

A thesis

presented to

the faculty of California Polytechnic State University,

San Luis Obispo

In partial fulfillment

of the requirements for the degree

Master of Science in Mechanical Engineering

by

Jason Allan Smith

September 2011

COMMITTEE MEMBERSHIP

TITLE:	California Polytechnic State University Wind Resource Assessment
AUTHOR:	Jason Allan Smith
DATE SUBMITTED:	September 2011
COMMITTEE CHAIR:	Patrick Lemieux, Ph.D., Associate Professor
COMMITTEE MEMBER:	John Ridgely, Ph.D., Associate Professor
COMMITTEE MEMBER:	Russell V. Westphal, Ph.D., Professor

ABSTRACT

California Polytechnic State University
Wind Resource Assessment

Jason Allan Smith

Wind resource assessment at California Polytechnic State University shows there is potential for wind power generation on Cal Poly land. A computational fluid dynamics model based on wind data collected from a campus maintained meteorological tower on Escuela Ranch approximately 5 miles northwest of campus suggests there are areas of Cal Poly land with an IEC Class III wind resource at a height of 80 meters above ground. In addition during the daytime when the campus uses the most energy there are large portions of land with annual average daytime wind speeds above 6.9m/s. These areas have been identified by analyzing the wind speed and directional data collected at the meteorological tower and using it to create the boundary conditions and turbulence parameters for the computer model. The model boundary conditions and turbulence parameters have been verified through comparison between data collected at Askervein hill in Scotland during the 1980's and the results of a simulation of Askervein hill using the same model. Before constructing a wind farm for power generation, additional meteorological towers should be constructed in Poly Canyon to further confirm the wind resource prediction.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

California Polytechnic State University (Cal Poly) is part of the 23 campus California State University system and is located in San Luis Obispo, CA. It is the second largest land-holding university in California with total holdings of 9,678 acres and an enrollment of 19,325 students as of fall 2009. There are many different clubs and programs on campus specifically focused on sustainability through actions such as composting leftover food from campus dining, installing motion sensitive outdoor lighting areas which dim when no one is in close proximity, and a student run and maintained organic farm. In addition to the many clubs and programs devoted to being environmentally friendly, the recently completed Poly Canyon Village housing project on campus is LEED gold certified and buildings that are currently under construction will apply for LEED certification once completed. To continue with their commitment to sustainability Cal Poly is currently investigating the feasibility of building a wind farm on campus which could provide enough energy to offset all or part of the university's power consumption which ranges between 3 and 10MW depending on the time of day and year (Elliot 2010).

1.1 Problem Statement and Framework

The goal of this project is to determine the wind resource available to Cal Poly by means of collecting data from a university constructed and maintained meteorological (MET) tower and analyzing that data to determine the characteristics of the wind resource at that location, such as mean wind speed, max wind speed, and wind direction; then, evaluate the characteristics of the wind resource at the MET tower to determine if it is sufficient for power generation and if so, using computational fluid dynamics (CFD),

extrapolate the wind data subject to topography over the area of interest. Then use the extrapolated data to determine the best locations for installation of several utility scale wind turbines and additional MET towers to verify that the wind resource is as predicted. Acceptable locations for wind turbines are at least Class III which is equivalent to average winds of at least 6.9m/s at 80m.

To determine the optimal locations for wind turbines based on the wind resource, wind tunnel calibrations are performed on the anemometers used on the MET tower. The calibrations ensure the accuracy and validity of the data collected at the MET tower. The raw data is transmitted wirelessly every six minutes from the MET tower to a computer on campus where a custom MATLAB code is used to analyze the raw data collected at the MET tower. The code is run daily to process and archive the collected wind data. Once analyzed, the data is used to determine turbulence parameters and boundary conditions for a FLUENT computational fluid dynamics model. The model parameters are verified by comparing simulation results to collected data from a past study with similar geometry in Scotland (Taylor and Teunissen 1985). The results of the model may ultimately determine the optimal site for each wind turbine, or at least the location of additional MET towers to verify the wind resource.

Chapter 2: Review of Previous Wind Resource Assessment Studies

Wind resource assessment is a very time consuming process which requires extensive preparation and organization. First, sufficient meaningful data needs to be collected in an organized fashion which requires choosing the proper data measurement and storage equipment as well as the optimal location(s) to set up the equipment to collect the needed data. The choices of equipment used, location of equipment, and duration of data collection are different for each specific case yet often share significant similarities. Once sufficient data has been collected it needs to be filtered for anomalies and carefully analyzed to determine if further investigation of the wind resource is required and/or warranted. Last, once the questions of whether sufficient data has been collected and whether further exploration of the wind resource is warranted have been answered, creating a CFD model is a common means of extrapolating the data collected at a single or a few locations over a larger area of interest (Palma, et al. 2008).

Numerous studies have been carried out since the popularization of CFD modeling to investigate the different aspects of CFD model refinement such as; mesh structure and refinement, the modeling of boundary conditions, and different turbulence models and parameters to increase the accuracy of models with complex geometry and flow fields. Specifically models created for wind resource assessment have become popular for the purpose of advancing the field of wind energy. The many investigations of CFD model refinement have produced a great deal of agreement with regard to the aspects listed above which will be discussed in further detail later.

2.1 Data Collection, Filtering, and Analysis: Case Studies

The equipment used to measure the wind for each case study varies but usually includes anemometers and sometimes a combination of sonic detection and ranging (SODAR) or light detection and ranging (LIDAR) with anemometers. In addition to the equipment used to measure the wind, the frequency of measurement and time period over which the measurements are averaged during data analysis also varies.

One case study performed in the foothills of the Caha and Sheedy mountain ranges of Ireland uses 3 MET towers each supporting an anemometer and wind vane (Bechrakis et al. 2004). This study is unusual in that one MET tower was 30m tall and the other two were 40m tall; usually when multiple data collection sites are used the measurement height remains constant. Although the measurement heights of the towers were unusual the sample rate of the study was one second and the averaging period was the industry standard 10 minutes. Data was collected for one year and was stored on site by the data logger using a non-volatile memory card and transmitted via a GSM cell phone system to a PC daily.

In contrast with the study by Bechrakis et al., another study performed by a master's student in Idaho used a combination of one MET tower collecting anemometer data and a SODAR unit (Russell 2009). Russell obtained the anemometer data from an 80m tall MET tower managed by the Idaho National Laboratory (INL) and the SODAR unit was the Triton model made by Second Wind. The MET tower in this study was different from the study by Bechrakis et al. because the tower managed by the INL had anemometers at two different heights up the length of the tower, 30 and 80m. The study by Russell was similar to that of Bechrakis et al. with regard to the data averaging period

of 10 minutes and the means of storing and transmitting the data using an on-site data logger and a cellular phone based transmission device.

2.2 Computational Fluid Dynamics Models: Case Studies

Computational fluid dynamics models are great tools for wind resource assessment because they provide a means of extrapolating wind data collected at discrete locations over a broader area. This can require many millions of calculations and solving multiple complex equations simultaneously which is not possible to do by hand. However although CFD allows for modeling complex situations, the parameters of the model must be carefully controlled for accurate results. Two of the most important aspects of any CFD model are the mesh and the boundary conditions.

2.2.1 Mesh Structure and Refinement

The mesh is a means of discretizing the model geometry to solve the Navier-Stokes equations which are the foundation of most CFD software. A CFD model's mesh structure is the first thing to be defined once the geometry is established and its importance to model accuracy cannot be overlooked. To ensure a quality mesh it is necessary to establish a balance between high spatial resolution which is desirable for an accurate solution and lower resolution which requires less computing resources. In addition a structured mesh will require less computing power than a mesh composed of tetrahedral elements but is not as well suited for complex geometry. There are a number of different techniques used to establish the best possible mesh for a given amount of computing power. One common practice is to vary the spatial resolution of the mesh throughout the model in a controlled manner. This allows for higher resolution in places of interest such as near the ground surface within the atmospheric boundary layer. The

hub height of a wind turbine lies in this region. This technique saves on computation time by allowing lower resolution where gradients in the parameters of interest are likely to be much smaller.

For simplistic geometry like a two dimensional sinusoidal representation of a hill, a structured mesh is a good option (Griffiths and Middleton 2010). Figure 1 below shows the meshes Griffiths used for investigation of flow separation over a hill using Regional Atmospheric Modeling Simulation (RAMS) and FLUENT CFD software. Only the mesh used in the FLUENT simulation is relevant to this discussion. The body fitted mesh used for the FLUENT simulation is composed of structured non-orthogonal elements and employs the technique of using higher resolution near the ground surface and lower resolution in the far field. The mesh created for the RAMS simulation is much more simplistic because of limitations on user controllability in the RAMS meshing software.

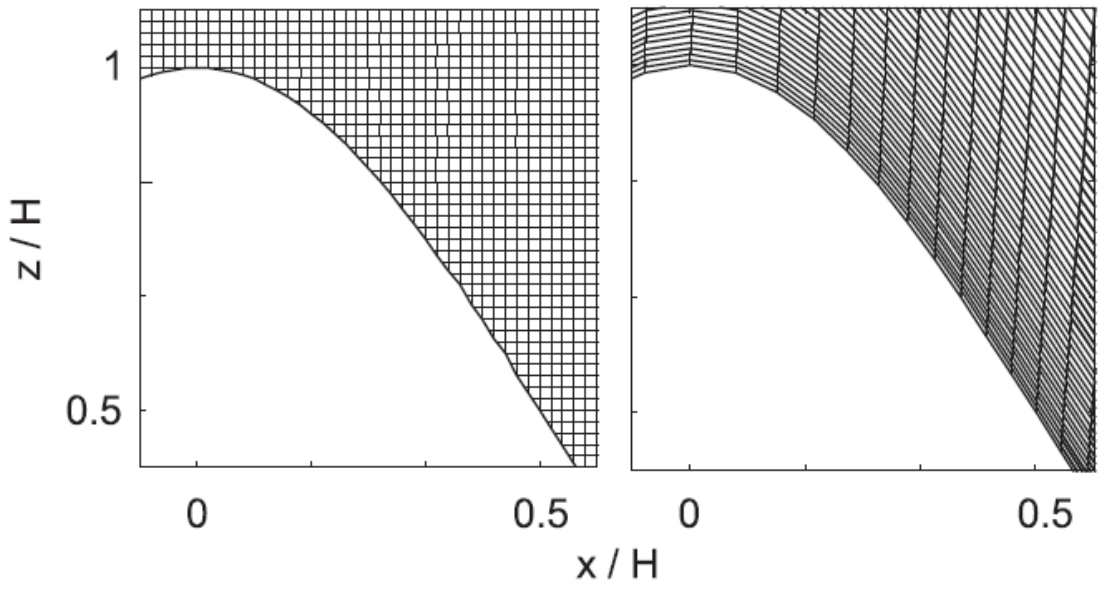


Figure 1: Two examples of structured meshes. The left hand image corresponds to a mesh created by RAMS software and the image on the right depicts the mesh used for a FLUENT CFD simulation.

A three dimensional example of a structured mesh was used to model a mountainous area near Beijing and highlights why structured meshes are not always desirable for complex terrain (Li et al. 2010). The area of interest is approximately 4km x 4km in the horizontal plane and approximately 2.6km tall. The mesh used to discretize this domain is composed of body fitted hexahedral elements and is shown in Figure 2 below. As is common practice grid inflation is used in this study. In grid inflation the vertical extent of a cell is increased by a small amount from the cell vertically adjacent to it. This study employs an inflation ratio of 1:1.02 to provide higher resolution near the ground surface. There are 40 layers of elements between the ground surface and the top surface of the domain which is at a fixed elevation. This means that the height of each column of elements varies yet the inflation ratio between layers of elements is maintained throughout the model which means that the height of the first cell (the one closest to the ground) varies throughout the domain by as much as 35%. This phenomenon is then amplified when moving vertically away from the ground surface due to the fixed inflation ratio. It is usually best practice for the elements in the area of interest, in this case the first 200m or so from the ground, to be as uniform in size as possible. The reasons the authors settled for a mesh with widely varying first cell sizes are unclear from their published results.

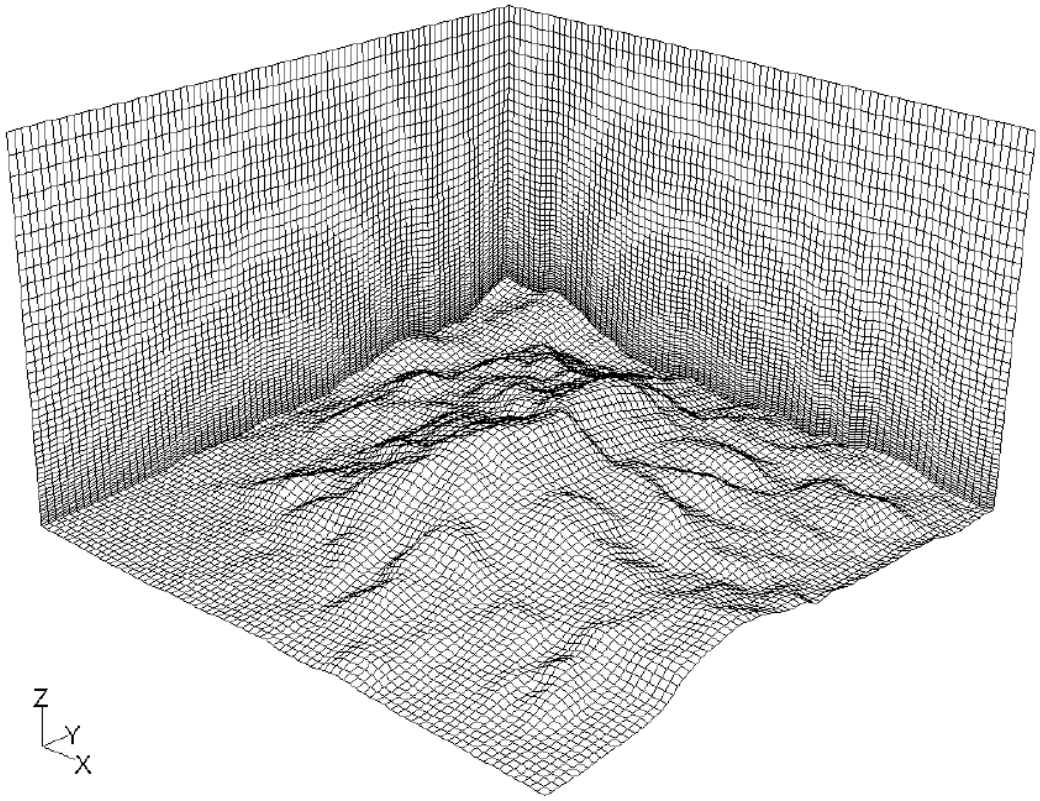


Figure 2: The mesh used by Li et al. to discretize an area composed of mountainous terrain northwest of Beijing.

For wind resource assessment an ideal mesh has the following qualities (ME 554 2011). First, it is always preferable to have a structured mesh over a mesh composed of tetrahedral elements to reduce the computing power needed. Second, it is desirable to have high resolution in the vertical plane within the atmospheric boundary layer where velocity gradients are the largest. Third, best results are obtained if the first cell height and inflation ratio are maintained throughout the model within the atmospheric boundary layer. Last, the horizontal spatial resolution throughout the model must be fine enough to capture the true behavior of the wind. Many of these parameters are highly dependent on the model geometry and flow conditions and therefore need to be determined through a process of trial and error called a grid independence study. This process is discussed in further detail later.

2.2.2 Boundary Conditions

Once a suitable mesh has been generated for the domain of interest the boundary conditions of the model need to be specified. Most CFD programs include a wide variety of boundary conditions that are available for use in which the user sets a few parameters to define the boundary condition on the entire surface. In addition, most programs also give the user the option to define their own boundary conditions or import boundary condition data from other software programs. This allows the user to have much more control when specifying the boundary conditions. This can provide more accurate simulation results, especially when dealing with complex models because the boundary condition has more flexibility. The user defined boundary condition properties can vary more along the surface than a boundary condition defined by setting a few parameters.

Wind resource assessment studies usually involve a domain that is roughly a rectangular prism except the bottom surface represents the topography of the area of interest. For the purpose of discussing boundary conditions, one surface perpendicular to the topography (ground) surface will be considered the inlet surface which corresponds to the plane where the incoming wind blows through first, the surface directly opposite the inlet surface will be called the outlet surface, the remaining two surfaces perpendicular to the topography surface will be called the side surfaces, and the surface parallel to the topography surface will be called the top surface.

In a study performed by the Centre for Renewable Energy Sources (CRES) for the purpose of wind turbine siting in complex terrain, a CFD code which was developed by CRES was run with the following boundary conditions (Politis and Chaviaropoulos 2008). The inlet surface was specified as a log law velocity profile.

$$u = \frac{u^*}{\kappa} \ln\left(\frac{z}{z_o}\right) \quad [1]$$

Where u represents the velocity in the streamwise direction at height z, κ is the Von-Karman constant, z_o represents the roughness length, and u_* is the friction velocity.

$$u^* = \sqrt{\frac{\tau_w}{\rho}} \quad [2]$$

Where τ_w is the shear stress at the surface and ρ is the fluid density. Neumann conditions were applied to the outlet and side surfaces, the no slip condition was applied to the ground surface, and the velocity components were specified for the top surface as u_x=1, u_y=u_z=0 where x was the streamwise direction. A Neumann or symmetry boundary condition stipulates that there are no parameter gradients across the boundary such as velocity or pressure.

A different study investigating the modeling of wind farms in both flat and complex terrain used similar boundary conditions (Prospathopoulos et al. 2010). In this study the CFD model was created in Ansys FLUENT and run with an inlet surface specified as a log law velocity profile, Neumann conditions for the side surfaces, and the no slip condition along the ground surface. These boundary conditions were all identical to the CRES study but different boundary conditions were used for the top and outlet surfaces. In this study the top surface was modeled using Neumann conditions and the outlet was specified as a pressure outlet with the pressure set to atmospheric pressure. It is difficult to quantify how these changes in boundary conditions affect the simulation results because the topography and flow fields are different.

Another common inlet boundary condition used for CFD models is the specification of a power law velocity profile rather than a log law profile (Russell 2009). The power law velocity profile is defined as follows.

$$u = u_o\left(\frac{z}{z_o}\right)^{\alpha} \quad [3]$$

Where u represents the velocity in the streamwise direction at height z, u_o is a reference velocity at height z_o, and α is the wind shear exponent. When wind shear data has not been measured it is customary to use α=1/7 for the power law profile. The use of this velocity profile requires knowledge of the wind speed at a reference height z_o, if wind speeds are known at multiple heights this equation can be rearranged to solve for α to provide a more accurate velocity profile.

$$\alpha = \frac{\ln\left(\frac{u_2}{u_1}\right)}{\ln\left(\frac{z_2}{z_1}\right)} \quad [4]$$

In the previously described study of the wind resource in a mountainous region near Beijing the boundary condition data for FLUENT came from the results of the meso-scale meteorological model called RAMS (Li et al. 2010). This allowed for more specific control of the boundary conditions. The trend of using the output from a meso-scale meteorological model to define boundary conditions for CFD models is becoming more popular because of the enhanced specificity of the boundary conditions that can be created in this manner. Another popular meso-scale atmospheric modeler similar to RAMS is called Weather Research Forecasting (WRF).

There are advantages and disadvantages to each of the different boundary conditions. For example, sometimes a log law velocity profile can be a more accurate inlet boundary condition than a power law profile because it takes the ground cover into account through the roughness length z_o. However if the ground cover changes significantly throughout the model domain a power law velocity profile may be more accurate. When using a symmetry (Neumann) boundary condition a significant assumption is being applied to the model. It is assumed that there are no gradients in velocity, pressure, etc. across the boundary. Depending on the flow conditions and the extent of the model domain this could be an acceptable assumption or it could be a terrible one. The boundary condition most used in place of a symmetry condition is the pressure outlet. This might allow for the extent of the model domain to be smaller because the pressure gradient across the boundary doesn't need to be zero. However, if a pressure outlet is used an appropriate back pressure must be specified or else the results of the simulation will not be accurate. Last, user defined boundary conditions based on the output of other modeling software can be very spatially specific but can be meaningless if it is based on an inaccurate model. Since all boundary conditions are tradeoffs it is imperative that any CFD model is validated before trusting the solution results.

2.3 Turbulence Models and Parameters

Turbulence modeling is one of the most important and least understood aspects of CFD modeling. It is essential to model the wind as turbulent rather than laminar for many reasons. First, there are many natural and man-made physical obstacles which introduce turbulence into the wind such as trees, terrain irregularities like cliffs, buildings, and even

wind turbines. As the wind flows over, past, or through these obstacles an otherwise uniform flow field is disrupted. Second, wind is a fluctuating phenomenon. Wind changes both speed and direction frequently and without a distinct pattern. Last, temperature gradients can create turbulence through natural convection. Measuring turbulence requires equipment capable of detecting very small fluctuations in the velocity AND direction of the wind on very short time scales. In addition as mentioned above, turbulence does not follow a distinct pattern and is always changing. For all these reasons turbulence modeling is difficult but is of extreme importance for accurate model results.

The Mach number is a means of quantifying a fluid's compressibility. A fluid with a Mach number greater than 0.3 is considered compressible. A Mach number less than 0.3 is considered incompressible. All but the most extreme weather conditions in nature are considered incompressible because the flow speeds aren't high enough to yield a Mach number greater than 0.3. This is significant because the Navier-Stokes equations which are the backbone of most CFD software change significantly if compressibility is taken into account.

$$M = \frac{u}{c} \qquad [5]$$

Where M is the Mach number, v represents the flow speed, and c is the speed of sound. To compute the flow over geometry, CFD codes solve the conservation of mass (continuity) and conservation of momentum (Navier-Stokes) equations simultaneously. For incompressible flows with constant viscosity the continuity and Navier-Stokes equations are written as:

$$\nabla \cdot \vec{v} = 0 \qquad [6]$$

$$\rho \frac{D\vec{v}}{Dt} = \rho g - \nabla P + \mu \nabla^2 \vec{v} \qquad [7]$$

Where ∇ is the gradient operator, $\vec{v}$ is the velocity vector, ρ is the fluid density, g is gravity, P is pressure, μ is the fluid viscosity, and ∇^2 is the Laplacian operator.

The conservation of mass and conservation of momentum equations need to be time averaged before they can be used to model turbulence. To do this, each of the time fluctuating variables (u, v, w, and P) is often separated into a mean component plus a fluctuating component. In general, for an arbitrary property ϕ:

$$\phi = \bar{\phi} + \phi' \qquad [8]$$

Where $\bar{\phi}$ is the mean component of ϕ and ϕ' is the fluctuating component. This representation of each time fluctuating variable is then substituted back into its respective equation and then the equations are time averaged. The time average is computed by:

$$\bar{\phi} = \frac{1}{T} \int_{t_o}^{t_o+T} \phi dt \qquad [9]$$

Where in this case ϕ can either be a variable or an entire equation and T is the period of integration which must be large compared to the relevant period of fluctuations in the flow.

The time averaged form of continuity for incompressible flow can be written as:

$$\nabla \cdot \bar{v} = 0 \qquad [10]$$

Equation 10 shows that the gradient of the mean component of velocity is equal to zero. The time averaged Navier-Stokes equations which are often called the Reynolds Averaged Navier-Stokes (RANS) equations are written as:

$$\rho \frac{D\bar{v}}{Dt} = \rho g - \nabla \bar{P} + \nabla \cdot \tau_{ij} \qquad [11]$$

Where τ_{ij} is the Reynolds stress tensor which for Cartesian coordinates is 3X3 and has 6 independent terms.

$$\tau_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \rho \overline{u_i' u_j'} \qquad [12]$$

If the difference is taken between the Navier-Stokes and Reynolds Averaged Navier-Stokes equations the result is that the gradient of the fluctuating component of velocity is also equal to zero.

$$\nabla \cdot v' = 0 \qquad [13]$$

Together the time averaged continuity and RANS equations have more unknowns than equations and therefore form an open set of equations which CFD codes need to solve simultaneously. This is impossible without closing the equation set, turbulence models are a means for closing the set of equations without introducing additional unknown quantities. There are many different turbulence models that have been developed, two of the most commonly used models will be discussed in further detail below.

2.3.1 The *K-ε* Turbulence Model

The *K-ε* turbulence model is a two equation model meaning that it adds two partial differential equations (PDE's) to the set consisting of continuity and the RANS equations to close the set. This model accounts for the transport of turbulent kinetic energy and for turbulent dissipation. This model is often used for wind resource assessment because it is particularly accurate for free shear layer flows with small pressure gradients. There are three variations of this model the Standard, Realizable, and RNG *K-ε* models. The Standard and Realizable models are used most often for wind resource assessment. The Standard model will be outlined in the following discussion.

Two transport equations are added to the equation set, one for turbulent kinetic energy and one for dissipation.

$$\frac{DK}{Dt} \approx \frac{\partial}{\partial x_j}\left(\frac{\upsilon_t}{\sigma_K}\frac{\partial K}{\partial x_j}\right) + \nu_t \frac{\partial \overline{u_i}}{\partial x_j}\left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}\right) - \varepsilon \qquad [14]$$

$$\frac{D\varepsilon}{Dt} \approx \frac{\partial}{\partial x_j}\left(\frac{\upsilon_t}{\sigma_\varepsilon}\frac{\partial \varepsilon}{\partial x_j}\right) + C_1 \nu_t \frac{\varepsilon}{K}\frac{\partial \overline{u_i}}{\partial x_j}\left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}\right) - C_2 \frac{\varepsilon^2}{K} \qquad [15]$$

Where the turbulent viscosity can be calculated by:

$$\nu_t \approx \frac{C_\mu K^2}{\varepsilon} \qquad [16]$$

$$K = \frac{1}{2}\left(\overline{u_i' u_i'}\right) \text{ or } K \approx \frac{3}{2}\left(UI\right)^2 \qquad [17]$$

Where the second formulation is a means of approximating *K* with more easily measured parameters. *U* is the velocity magnitude and *I* is the turbulence intensity. The

turbulence intensity is equal to the standard deviation of wind speed measurements divided by the mean of those measurements. Then the dissipation can be calculated from the turbulent kinetic energy.

$$\varepsilon \approx \frac{C_{\mu}^{3/4} K^{3/2}}{0.07L} \quad [18]$$

Where L is a characteristic length based on the flow and geometry. These equations include five empirical constants C_{μ}, C_1, C_2, σ_K, and σ_{ε}. These constants are not universally agreed upon but the most popularly used values are C_{μ}=0.09, C_1=1.44, C_2=1.92, σ_K=1.0, and σ_{ε}=1.3 (Launder and Spaulding 1974). This turbulence model is used in many wind resource assessment studies, Russell (2009) ran simulations using the K-ε model using the two sets of coefficients and compared the results. One determined by Launder and Spaulding and the other developed by Alinot and Masson (2005) where C_{μ}=0.03329, C_1=1.176, C_2=1.92, σ_K=1.0, and σ_{ε}=1.3.

2.3.2 The *K*-ω Turbulence Model

The K-ω turbulence model is a two equation model as well. This model accounts for the transport of turbulent kinetic energy and for specific dissipation. There are three variations of this model the Wilcox, Modified Wilcox, and SST K-ω models however, the Wilcox model is used most often for wind resource assessment and will be outlined in the following discussion.

Two transport equations are added to the equation set, one for turbulent kinetic energy and one for specific dissipation.

$$\frac{DK}{Dt} \approx \tau_{ij} \frac{\partial u_i}{\partial x_j} - \beta^* K\omega + \frac{\partial}{\partial x_j}\left[\left(+ \sigma^* \nu_t \right) \frac{\partial K}{\partial x_j} \right] \quad [19]$$

$$\frac{D\omega}{Dt} \approx \alpha \frac{\omega}{K} \tau_{ij} \frac{\partial u_i}{\partial x_j} - \beta\omega^2 + \frac{\partial}{\partial x_j}\left[\left(+ \sigma \nu_t \right) \frac{\partial \omega}{\partial x_j} \right] \quad [20]$$

Where υ is the kinematic viscosity, τ_{ij} was defined above in Equation 12, v_t is the ratio of K to ω, and α=5/9, β=3/40, β^*=9/100, σ=1/2, and σ^*=1/2 are all empirical constants (Wilcox 1988). The K-ε turbulence model is used for the Cal Poly wind resource assessment because it is well suited for the free shear flows present in atmospheric flow conditions (ANSYS 2006).

Chapter 3: Poly Canyon Wind Resource Assessment

The proposed site for the wind farm on the Cal Poly campus is located in a region consisting of a series of hills and valleys known as Poly Canyon. The ground cover in the area mostly consists of short to medium length grasses ranging between 6 and 18 inches in length as well as a few rocks, bushes, and trees. A picture of the area is shown in Figure 3 below.

Figure 3: The right half of this image is the proposed area for the wind farm which contains multiple hills and valleys complicating the prediction of the wind resource.

Poly Canyon lies just north of the campus core and is accessible via a dirt road from campus that loops around to California Highway 1 as shown in Figure 4. The area inside the red box is the main campus core. The yellow line represents a dirt road which travels through Poly Canyon connecting the campus core to California Highway 1. The area enclosed by the road through Poly Canyon is owned by Cal Poly and is under consideration for the wind farm.

Figure 4: A Google Maps image of San Luis Obispo, Cal Poly, and Poly Canyon.

3.1 Wind Resource Data

In order to map the wind resource in Poly Canyon successfully there are a few main aspects of the wind that need to be quantified. First and most importantly, the speed at which the wind blows needs to be accurately measured over the course of one full year at a minimum. Second, the direction the wind is blowing from needs to be examined. Last, the change in wind speed as a function of height off the ground needs to be explored to give an idea of how the wind speed measurements will scale when interpolating to the wind turbine hub height.

3.1.1 Data Collection

To facilitate collection of the data required for the wind resource assessment of Poly Canyon a MET tower was erected. The 80 foot tall tower was built on top of a ridgeline in Escuela Ranch approximately 5 miles northwest of Poly Canyon and has an anemometer every 20 feet up the length of the tower. It is essential to measure the wind speed at multiple heights off the ground to determine the wind shear and velocity profile. Both the terrain and ground cover in Escuela Ranch are comparable to Poly Canyon which is important to ensure the collected data is meaningful. Figure 5 shows the spatial relationship between Escuela Ranch and Poly Canyon. The blue x in the figure represents the location of the MET tower in Escuela Ranch a few miles north of Poly Canyon on highway 1. Rotating cup anemometers on the MET tower are used to measure the wind speed however, the wind speed isn't measured directly. The number of revolutions the cup anemometer makes every ten seconds is recorded on site and then is later converted off site to a wind speed. The number of revolutions the anemometer makes in each ten second interval is counted and recorded using a custom data acquisition device called a

Swoop board. The Swoop board was designed by John Ridgely Ph.D., a professor at Cal Poly and a member of the thesis committee. In addition to the anemometers, at the 60 foot tower height a wind vane is installed to measure the direction from which the wind is blowing in the form of sine and cosine values. The direction data is also recorded with the Swoop board. The ten second interval wind and direction data is averaged over six minute periods and then transmitted wirelessly to a computer on campus via a Digi MaxStream 9XStream 900MHz wireless transceiver. There is a wireless transceiver at each tower height and they are set up in a wireless mesh network so all of the signals are routed through whichever transceiver has the best connection with the receiver on campus. A custom C program is running on the receiving computer to log each line of data as it is received. Once all of the data from the day has been received on campus it is processed by a custom MATLAB code to convert the number of anemometer rotations to a wind speed using the transfer functions determined during calibration of the anemometers in the Mechanical Engineering (ME) department's wind tunnel and the sine and cosine values corresponding to wind directions into a value in degrees between 0 and 360. The anemometer calibration process and the analysis of data using the MATLAB code will be discussed in further detail later.

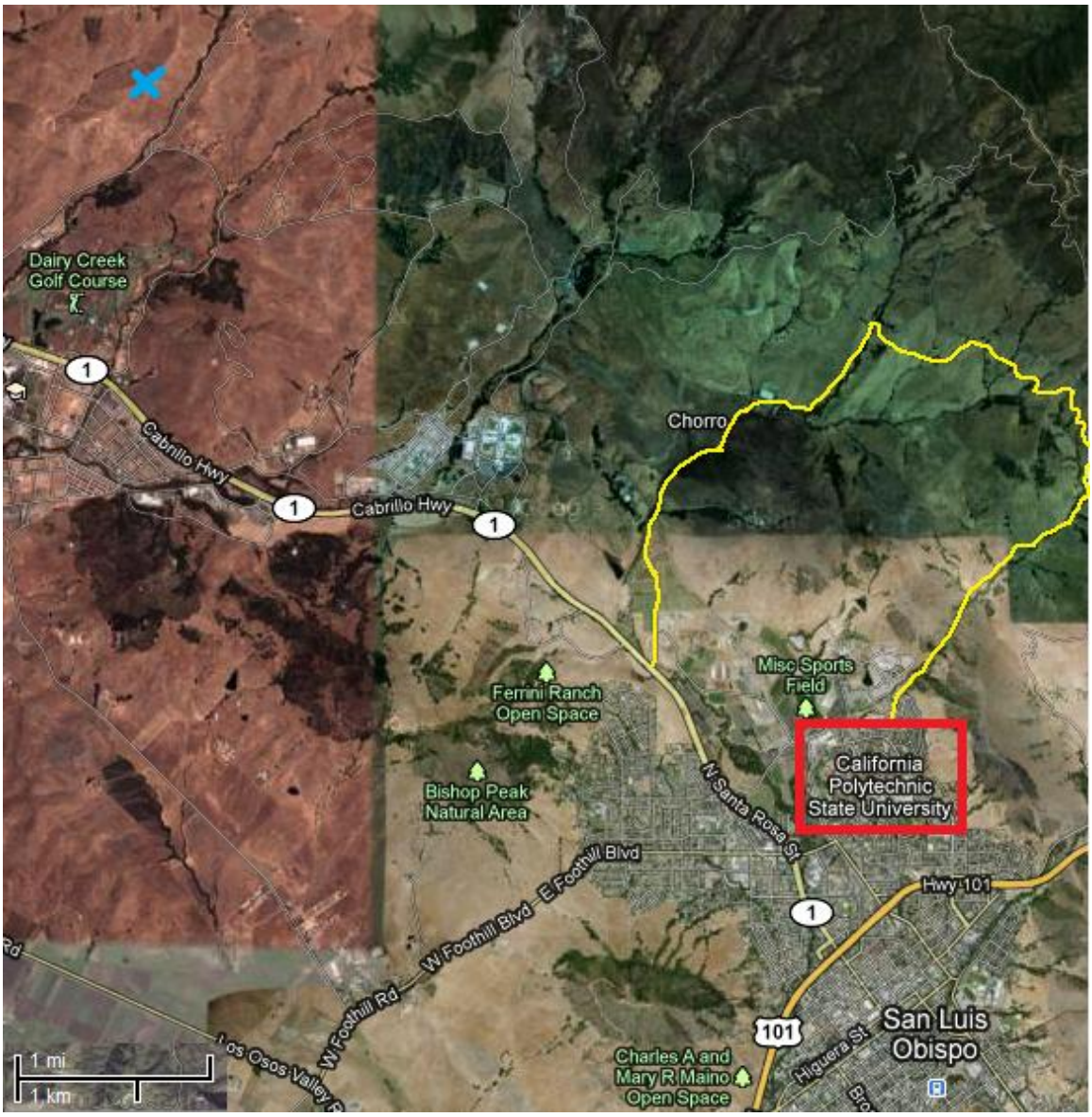


Figure 5: Google Maps image showing the spatial relationship between the MET tower on Escuela Ranch and Poly Canyon.

3.1.2 Overview of IEC Calibration Standard

The International Electrotechnical Commission (IEC) has a very specific standard regarding wind turbines and power generation (IEC 61400-12-1 2005). Annex F of this standard specifically refers to the calibration requirements, set up, and procedure for calibrating cup style anemometers. The following discussion outlines the main points of

the standard as well as whether the calibrations performed at Cal Poly meet those requirements.

The IEC standard outlines many requirements pertaining to the wind tunnel in which the anemometers will be calibrated. First, all measurement equipment used for calibrations must be traceable to the National Institute of Standards and Technology (NIST) and calibrated independently from any anemometers. Second, the blockage ratio which is defined as the frontal area of the anemometer compared to the wind tunnel cross section must be less than 0.05 for closed sections and 0.1 for open sections. Third, the flow within the wind tunnel must be uniform to within 0.2% in all three directions; transverse, axial, and longitudinal. Last, a reference anemometer must be designated. This anemometer is only used as a reference for wind tunnel calibrations and is never used in the field. In addition, anemometers calibrated other than the reference anemometer must undergo round robin testing at other facilities which meet the IEC standard to ensure that calibrations differ by less than 1% over a range of 4 to 16m/s.

The Cal Poly wind measurement setup meets some of these requirements but not all of them. It is too expensive and/or time consuming to follow the all of the guidelines in the IEC standard. Some but not all of the measurement equipment used at Cal Poly is NIST traceable. For example the Swoop board used to measure anemometer rotation frequency is not NIST traceable. The ME departments wind tunnel has a 2ft by 2ft cross section while the anemometer and mount have a frontal area of $18.6in^2$. This equates to a 0.032 blockage ratio which meets the requirement for a closed test section. Flow uniformity within the Cal Poly wind tunnel was investigated in the axial and longitudinal directions but not the transverse direction. The flow uniformity was not within 0.2% in

either the axial or longitudinal direction. Last, Cal Poly has designated a reference anemometer for this project however it has not undergone round robin testing since the Cal Poly wind tunnel does not meet the IEC standard.

In addition to requirements regarding the wind tunnel, the IEC standard also outlines specific requirements for the set up and procedure used during anemometer calibrations. The anemometer must be mounted in the wind tunnel on a tube of the same diameter as it will be mounted in the field. After mounting the anemometer, the wind tunnel must be run for at least 5 minutes before calibration begins. This ensures that the effect of temperature variations on bearing friction within the anemometer is avoided. While calibrating an anemometer wind speeds must be chosen in an increasing and decreasing order to ensure hysteresis effects are not present. At each wind speed the wind tunnel must be allowed to reach steady state before any measurements are taken and measurements must be sampled at a rate of at least 1Hz and for an interval of at least 30 seconds. Calibrations performed at Cal Poly meet all of these requirements with the exception of the sampling frequency. The swoop board counts the number or revolutions of the anemometer every 10 seconds which equates to 0.1Hz however, data was sampled for at least 2 minutes at each wind speed.

Last, the IEC standard includes requirements for data and uncertainty analysis as well as reporting format. A linear regression performed on the data should yield a regression coefficient of $r>0.99995$. The uncertainty analysis is required to include both type A and type B uncertainty. Finally, the minimum required information for a calibration report includes a wind tunnel description, a sketch of the wind tunnel set up, flow quality measurements, measurement equipment calibration certificates, a detailed

procedure, repeatability documentation, and uncertainty analysis. None of these requirements are met during calibrations at Cal Poly. The linear regression coefficients are below the required limit, most likely because the facility does not meet all of the other strict requirements laid out in the IEC standard. Since many of the requirements for the standard are not met at Cal Poly the report format has not been followed and an uncertainty calculation has not been done since some equipment is not NIST traceable and has unknown uncertainty.

3.1.3 Cal Poly Wind Tunnel and Anemometer Calibration

To effectively and accurately calibrate the anemometers used on the MET tower for use in the field a wind tunnel is used to create a controlled environment in which the response of the anemometer can be observed and recorded. Before the anemometers can be calibrated, the wind tunnel needs to be calibrated. To ensure accuracy the wind tunnel must be calibrated using a NIST traceable measurement device. A NIST traceable cup style anemometer was purchased from NRG Systems for this purpose because it is nearly identical in size, geometry, and operation to the Second Wind anemometers used in the field. Specifically the geometry of the rotating cups is indistinguishable. A report summarizing the conditions and results of the calibration performed by OTECH Engineering Inc. for the NRG Systems anemometer can be found in Appendix A. The ME department's wind tunnel speed cannot be controlled directly, instead the rotation speed of the fan drawing air through the tunnel is controlled which in turn dictates the wind speed through the tunnel. The frequency of revolution of the NRG Systems anemometer was used to determine the relationship between the wind tunnel fan's rotation speed and the speed of the air moving through the wind tunnel. The anemometer

rotation frequency was calculated internally to the oscilloscope based on measurements of period. Five measurements were taken at each wind speed and averaged to provide a more robust calibration. To reduce any possible effects of hysteresis on the calibration, fan speeds were used in an increasing and then decreasing order (i.e. 5, 10, 15, 20, 17, 12, and 7Hz) rather than monotonically increasing or decreasing throughout the duration of the calibration. At each wind speed the wind tunnel was allowed to reach steady state before any measurements were taken. The data showing the relationship between fan rotation speed and wind speed for the ME department wind tunnel is shown below in Figure 6. The curve fit equation can be found in Appendix C. Although the data could be described with a linear fit, a fifth order fit was chosen to reduce the residuals below 1%, shown in Figure 7. It is necessary to reduce the residuals as much as possible because of the cubic relation between wind speed and power which is the driving force determining the feasibility of a wind farm in a wind resource assessment.

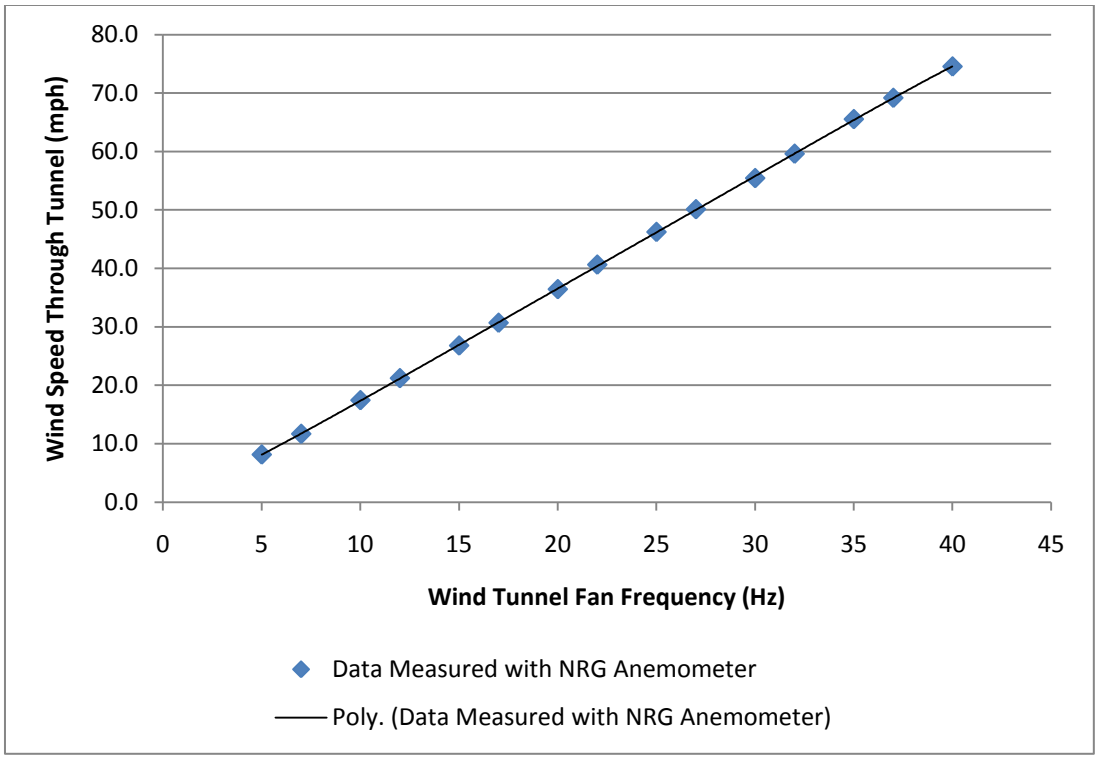


Figure 6: Data for the calibration of the wind tunnel found using the NIST traceable NRG Systems anemometer.

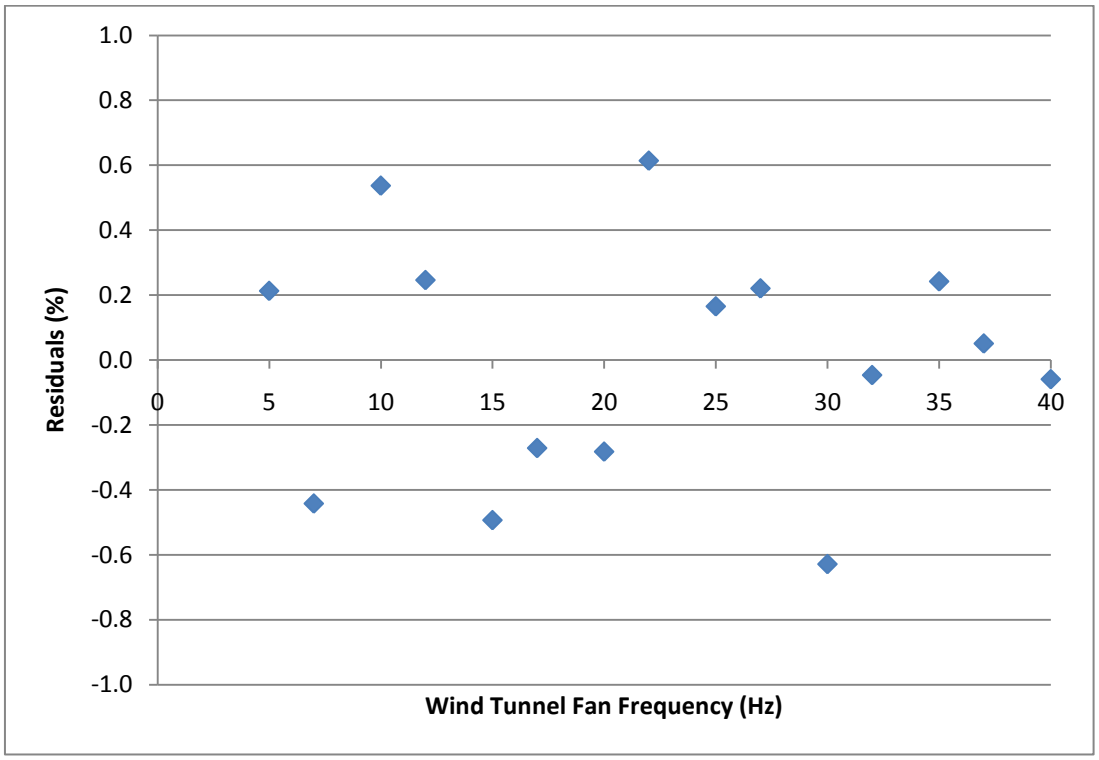


Figure 7: Residuals between the fifth order curve fit and measured data shown in percent of the measured value.

Once the wind tunnel was calibrated it was possible to begin calibrating anemometers. The anemometers on the MET tower were put into use before being calibrated in the wind tunnel so as to maximize the duration of data available for the wind resource assessment. Before calibration in the wind tunnel, the transfer function provided by Second Wind was used to convert revolution frequency to wind speed which can be found in Appendix B.

An identical Second Wind anemometer to the ones used in the field was calibrated in the wind tunnel to investigate the accuracy of the transfer function provided by Second Wind to determine if it was necessary to drop the MET tower to calibrate each of the anemometers used in the field. Dropping the MET tower to calibrate the anemometers used in the field would cause a large gap in the collected wind data which should be avoided if possible. When calibrating the anemometer the rotation was measured in two different ways simultaneously, with an oscilloscope and with a Swoop board. In the field the anemometers rotation frequency is only measured using the Swoop board so measuring the rotation with an oscilloscope as well provides a means to validate that the Swoop board is operating correctly. The results of the calibration are shown in Figure 8 below. There is good agreement between the rotation frequencies measured using the oscilloscope and Swoop board; the two frequencies differed by less than 2% over the entire range of wind speeds. The residuals for three different order curve fits applied to the Swoop board data are shown in Figure 9. The 6^{th} order curve fit was chosen because it provided the lowest residuals and it was only used within the range of data it is based upon. Ideally all of the residuals would be kept under 1% but that was not possible. The full curve fit equation can be found in Appendix C. After the calibration

curve was obtained it was possible to compare the transfer function provided by Second Wind to the one determined from wind tunnel calibration. This comparison shows at which wind speeds the transfer function provided by Second Wind is under or over predicting the actual wind speed. A plot of the error associated with using the Second Wind transfer function compared with the calibration curve determined in the wind tunnel for a given wind speed is shown in Figure 10.

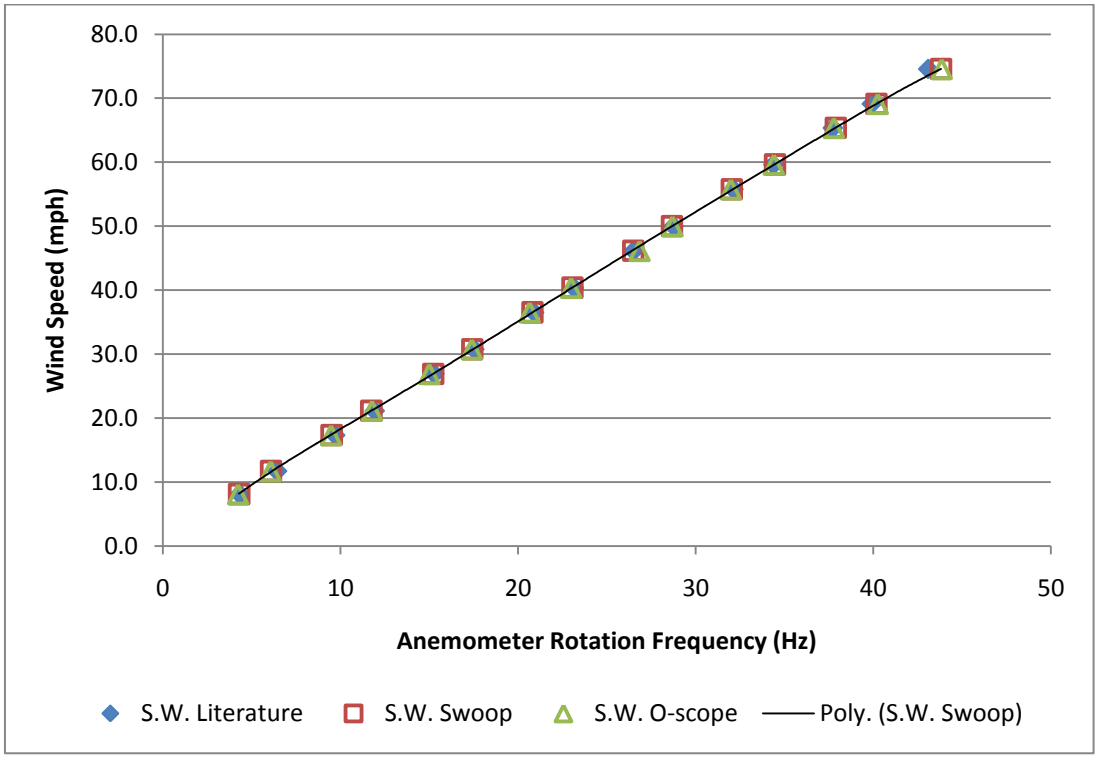


Figure 8: Data for the calibration of a Second Wind anemometer identical to those used in the field.

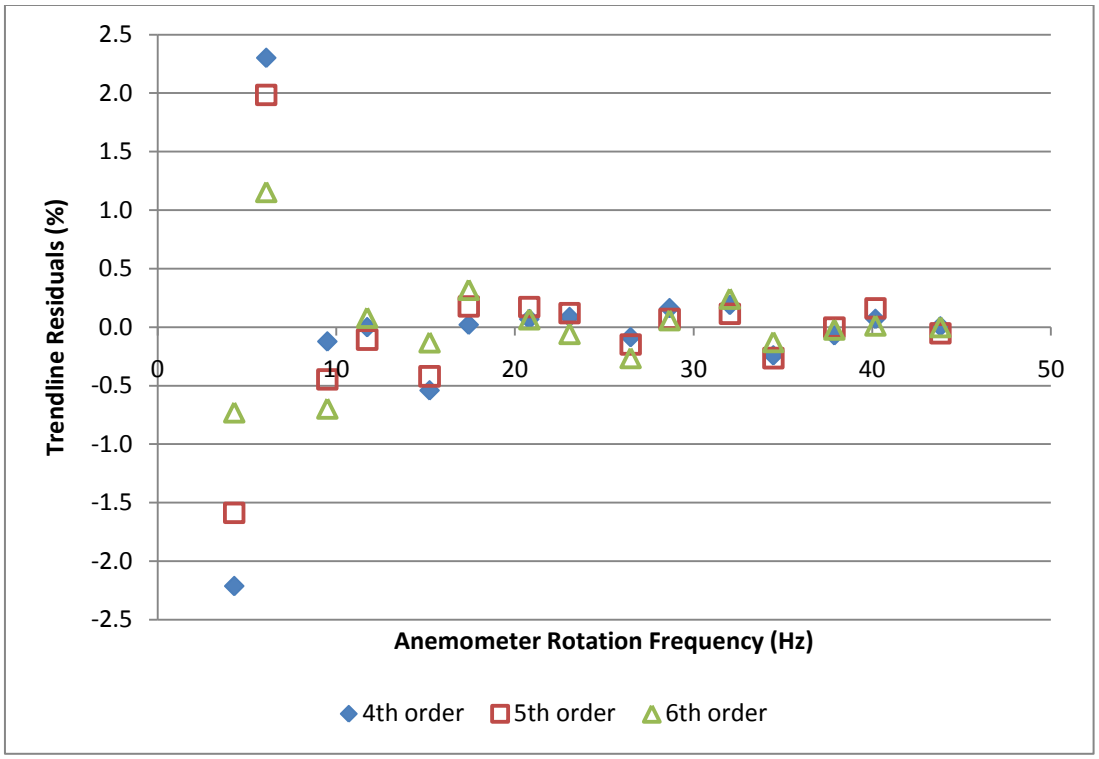


Figure 9: Residuals between 4th, 5th, and 6th order curve fits and the measured data shown in percent.

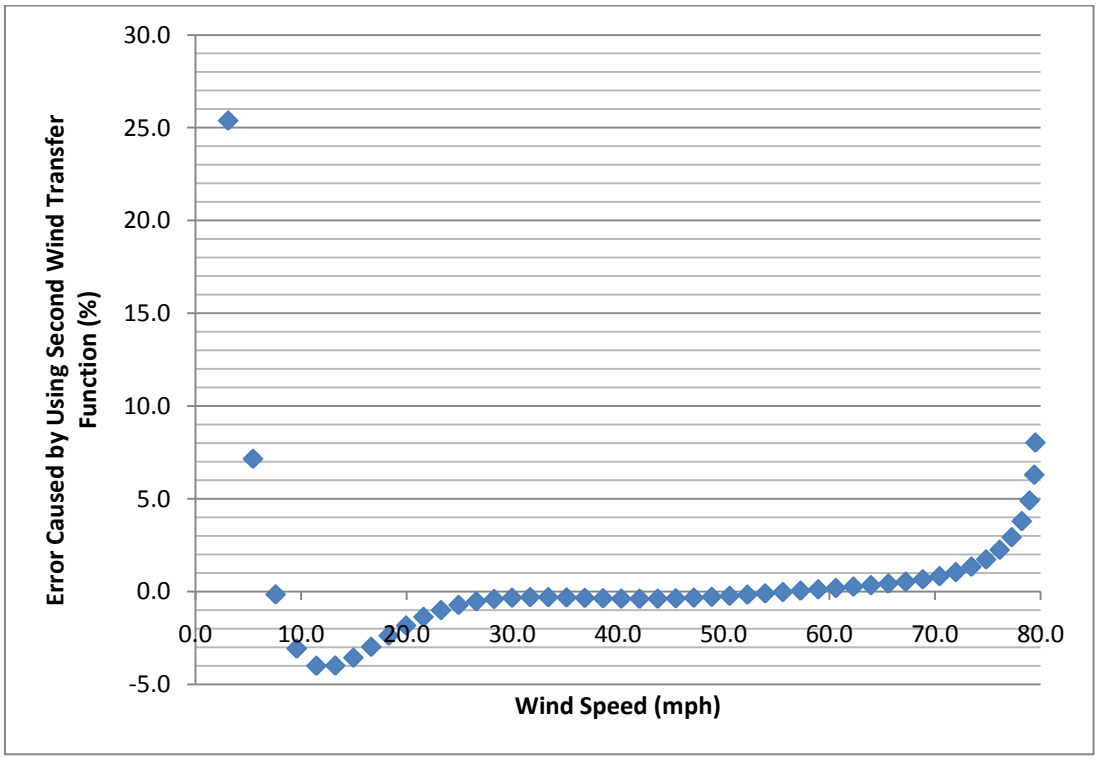


Figure 10: Plot showing how the transfer function provided by Second Wind, shown in Appendix B, over or under predicts the actual wind speed.

A positive error indicates the wind speed is over predicted while a negative error is indicative of an under prediction of the wind speed. The Second Wind transfer function greatly over predicts the wind speed at speeds below approximately 7 mph, under predicts the wind speed between 7 and 55 mph, and then over predicts the wind speed above 55 mph. This is significant for multiple reasons. First and most importantly it illustrates the need to calibrate each of the anemometers used in the field even though it will cause a gap in the collected data. In the present study care was taken to reduce error to under 1%. An over or under prediction of wind speed on the order of a couple percent has a much larger impact on the prediction of the power available in the wind at a given wind speed because power is proportional to velocity cubed. Second, this is encouraging because the Second Wind transfer function is under predicting the wind speed the most at speeds where a wind turbine will be operating most often. Last, it is acceptable that the Second Wind transfer function over predicts the wind speed below 7 mph and above 55 mph because these are approximately the cut in and cut out speeds of commercial wind turbines.

Due to the results of the initial anemometer calibration the MET tower was dropped on April 24, 2011 so that the field anemometers could be calibrated. After the anemometers were calibrated they were reinstalled on the MET tower and it was raised again on May 10, 2011. In total there was a 16 day gap in the wind data. Plots of the calibration data from each of the four field anemometers can be seen in Figures 11 through 14. In each of the figures the wind speed data is plotted in blue on the primary vertical axis while the residuals between the curve fit of appropriate order and wind

tunnel data are plotted in red on the secondary vertical axis. Each of the curve fit equations can be found in Appendix C.

The calibration data for the field anemometers at the 20 and 40ft tower heights was as expected; with 5th order curve fits the residuals were reduced to below 1%. The calibration data for the anemometer used at the 60ft tower height was not as linear as the data for the two anemometers at lower heights. The data for the 60ft anemometer required a 6th order curve fit and the residuals could only be kept within 2.5%.

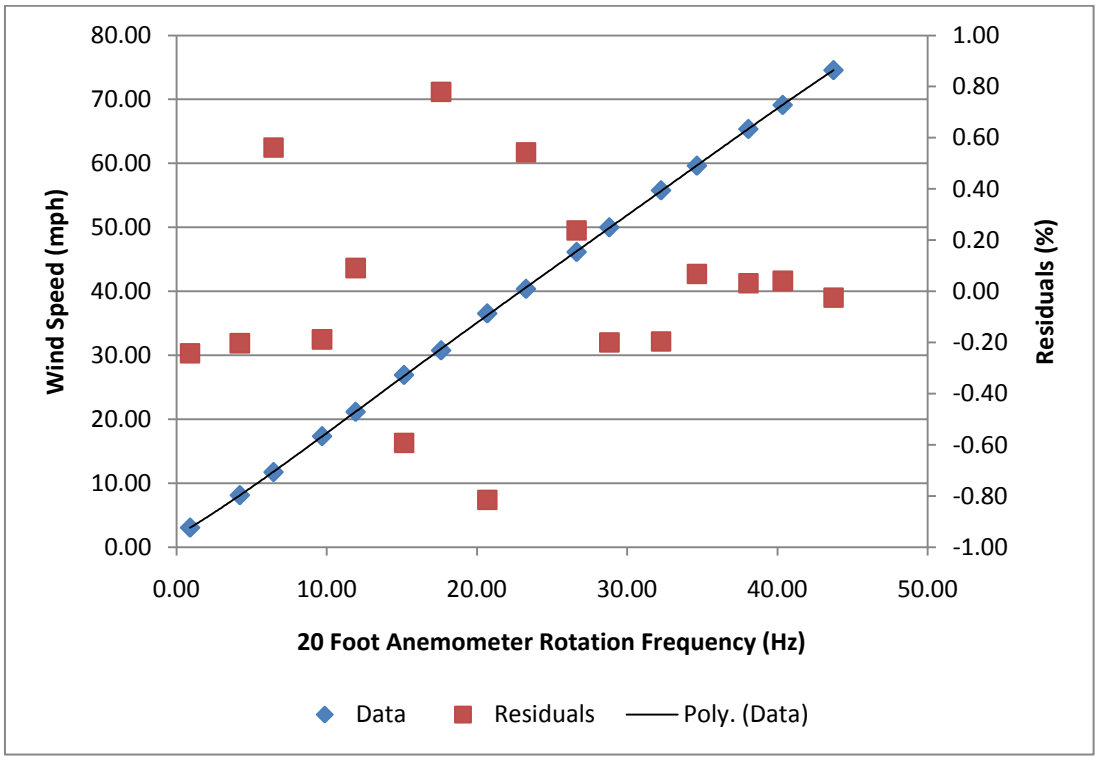


Figure 11: Calibration plot for the field anemometer at the 20ft tower height.

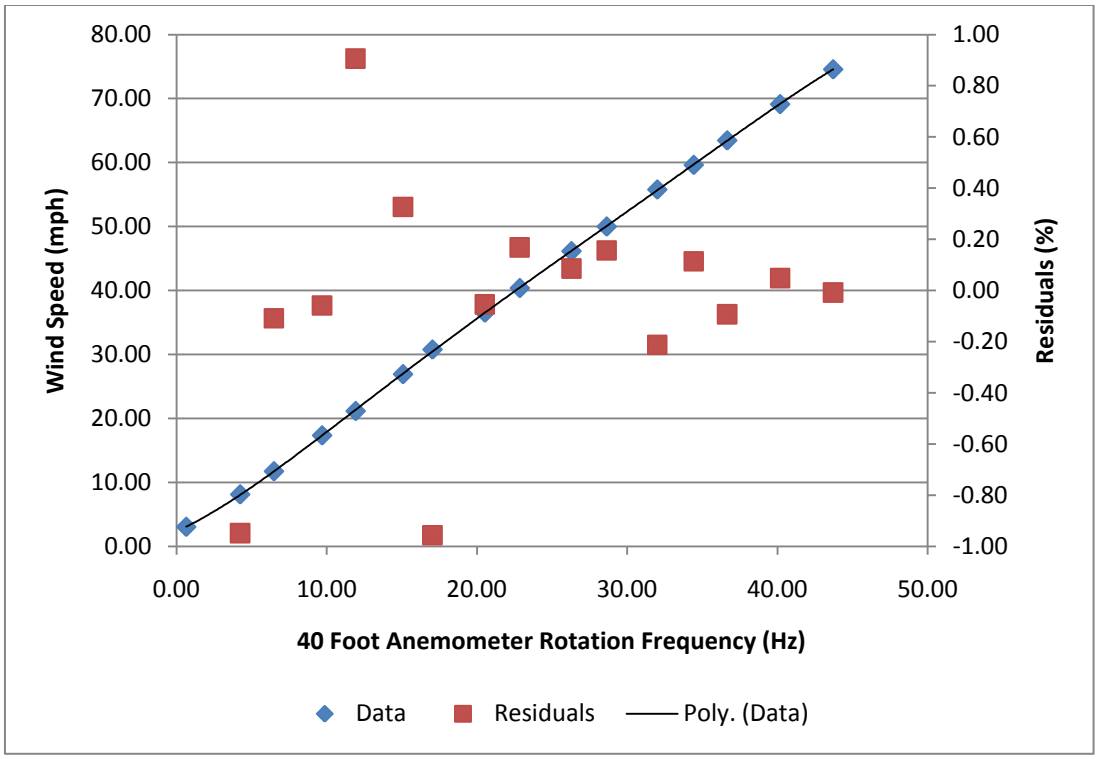


Figure 12: Calibration plot for the field anemometer at the 40ft tower height.

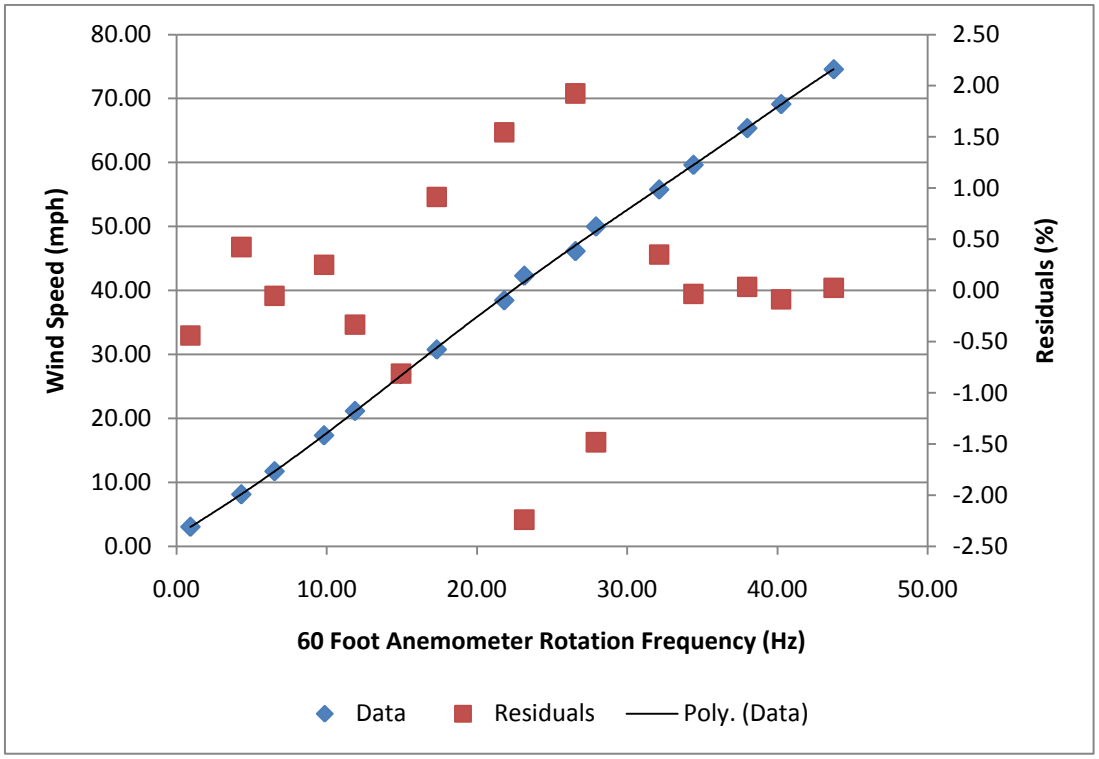


Figure 13: Calibration plot for the field anemometer at the 60ft tower height.

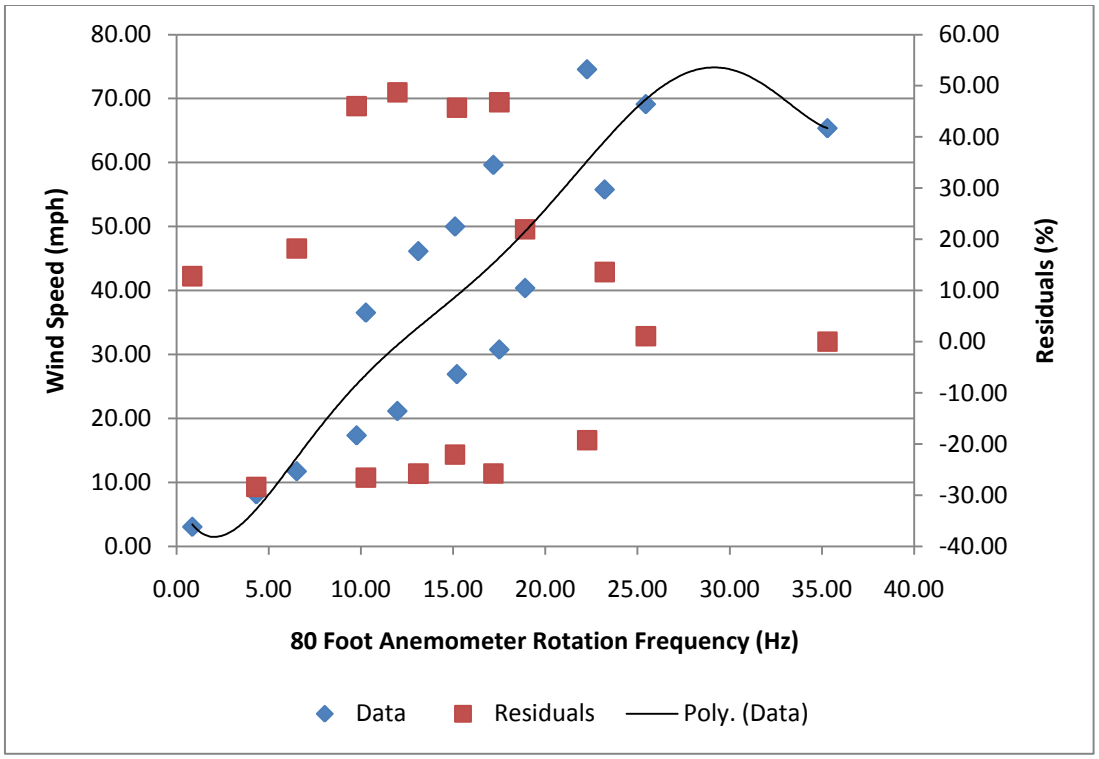


Figure 14: Calibration plot for the field anemometer at the 80ft tower height.

After calibration it was clear that the anemometer used at the 80ft tower height was malfunctioning and needed to be replaced. The anemometer functioned properly up to wind speeds of approximately 30mph but then the rotation frequency readings became highly erratic as shown in Figure 15.

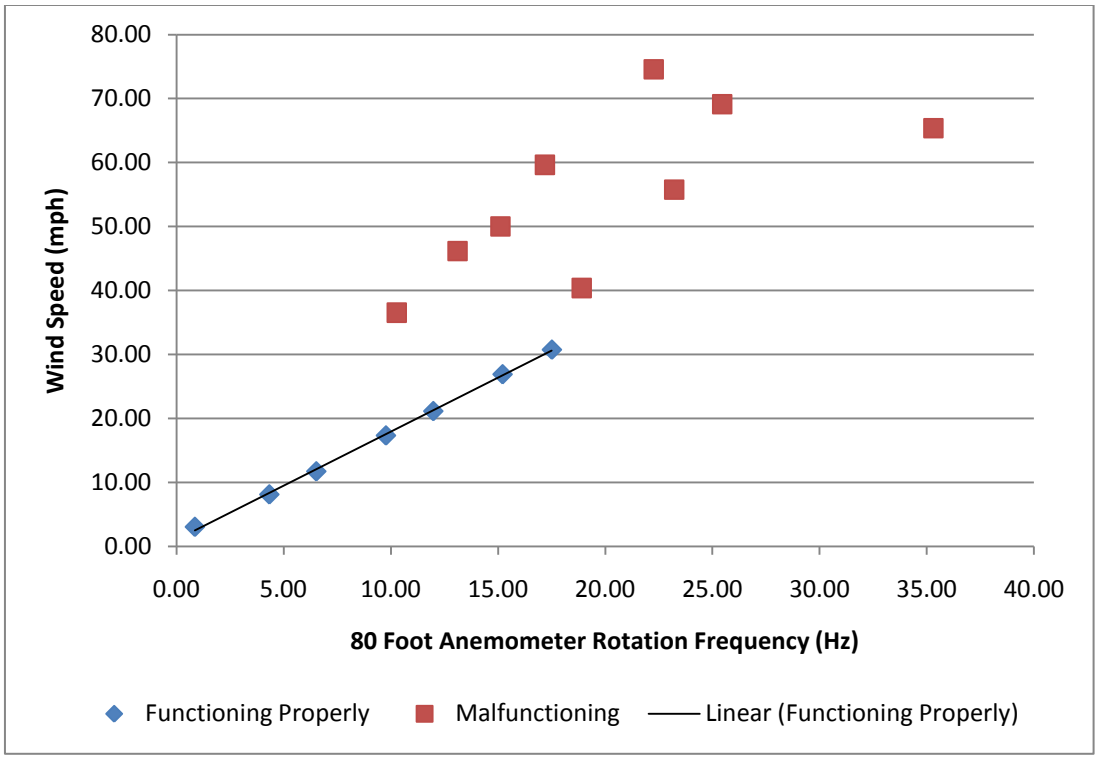


Figure 15: Calibration plot for the field anemometer at the 80ft tower height depicting at what wind speeds it was functioning properly and at what speeds it was malfunctioning.

When the MET tower was put back up the malfunctioning 80ft anemometer was replaced with the Second Wind anemometer that was calibrated initially (Figure 8). All data collected after the 80ft anemometer was replaced is processed using the calibration from the anemometer in Figure 8. All data that was collected before the malfunctioning anemometer was detected was reprocessed using a linear fit (shown in Appendix C) created using only the "Functioning Properly" data in Figure 15 (i.e. wind speed ≤ 30mph). This approach provides a best guess about the wind at 80ft but it must be noted that the data is suspect.

3.1.4 Data Analysis

Once the anemometers were calibrated it was possible to process the collected data and format it in a meaningful and insightful manner. The custom MATLAB code

was used to filter and plot the wind speed data. In addition, WindRose PRO software was used to process the wind speed and direction data from the 60ft tower height to create wind roses (WindRose PRO 2011).

The MATLAB code filters the data from each of the 4 tower heights and plots it by hourly average for 4 different durations; 1 day, 7 days, 30 days, and 365 days. For example, Figure 16 shows a plot created by the MATLAB code for the 365 days preceding June 5, 2011. In addition to plotting the hourly average wind speeds, the mean wind speed at each tower height over the entire duration is also shown.

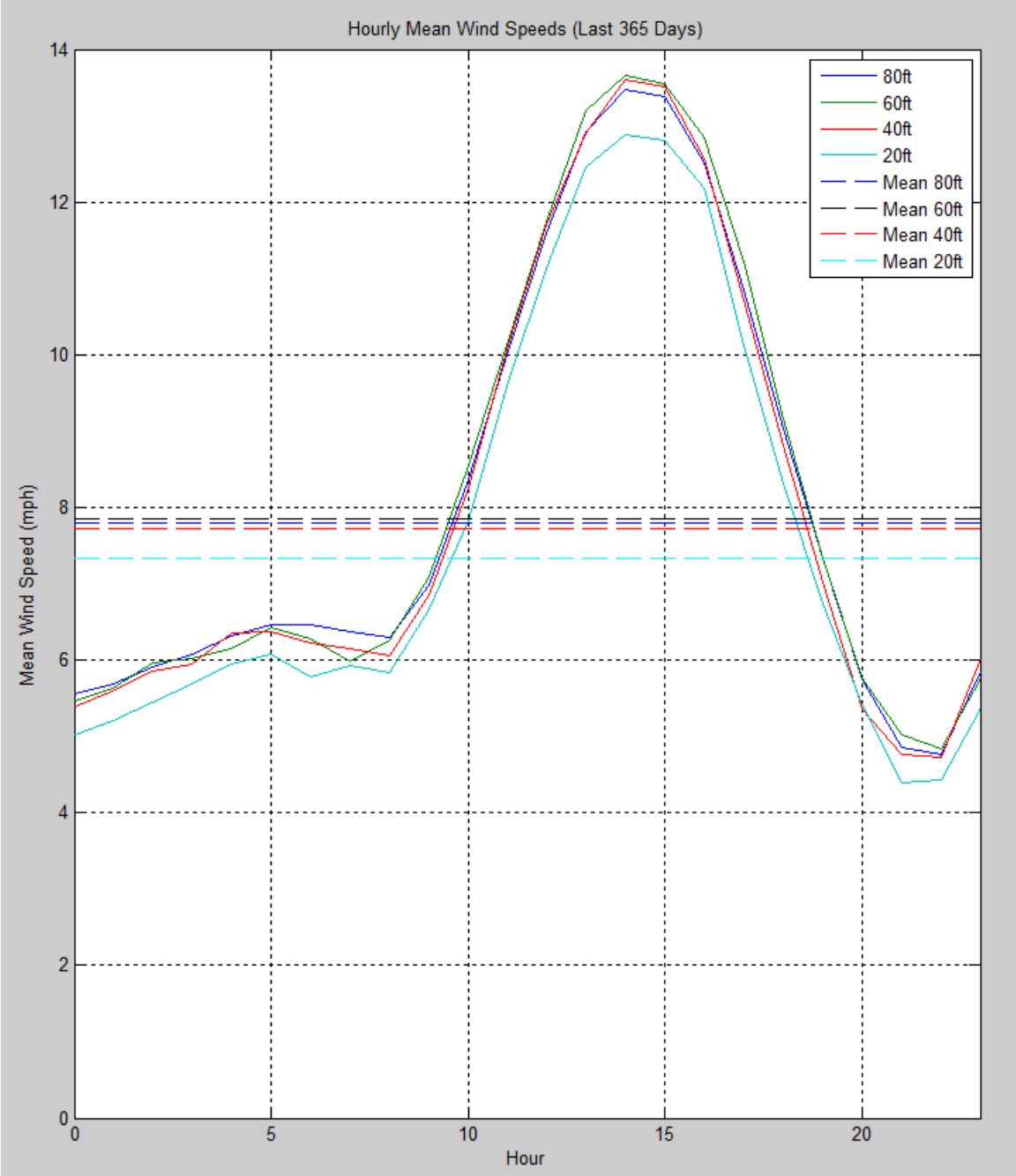


Figure 16: MATLAB plot of the hourly average wind speeds as well as the mean wind speed at each of the 4 tower heights for the 365 days preceding June 5, 2011.

This plot is very informative; it shows that there is a diurnal wind pattern with much stronger winds during the day than at night. This is encouraging because power consumption is much greater during the day as well. In addition this plot shows that the mean wind speed increases with tower height which is expected. Table 1 shows the

annual mean wind speed at each of the four MET tower heights, based on 24 hour averages.

Table 1: Annual mean wind speed for the 365 days preceding June 5, 2011.

MET Tower Measurement Height (ft)	Annual Mean Wind Speed (mph)	Annual Mean Wind Speed (m/s)
20	7.31	3.27
40	7.60	3.40
60	7.91	3.54
80	8.02	3.59

Based on the mean wind speeds at the different MET tower heights the shear exponent α can be calculated using Equation 4. Depending on which two MET tower heights are chosen the shear exponent changes, Table 2 summarizes the shear exponent calculated using each combination of MET tower heights. The shear exponents calculated from the MET tower data are very low. They are lower than what is often observed with smooth flat terrain and much lower than what is expected for terrain consisting of hills and valleys (Ray 2006). For this reason a shear exponent α=1/7 is used for the FLUENT simulations because it is considered the standard value for a power law velocity profile and is much closer to what is expected for hilly terrain than what is measured at the MET tower.

Table 2: Shear exponents calculated from the annual mean wind speeds at different MET tower heights.

MET Tower Measurement Heights (ft)	Shear Exponent, α
20 & 40	0.056
20 & 60	0.072
20 & 80	0.067
40 & 60	0.099
40 & 80	0.078
60 & 80	0.048

The other function of the MATLAB code is to plot Rayleigh and Weibull distributions based on the mean wind speed and standard deviation. Then those distributions are compared to histograms of the data at each tower height as shown in Figure 17.

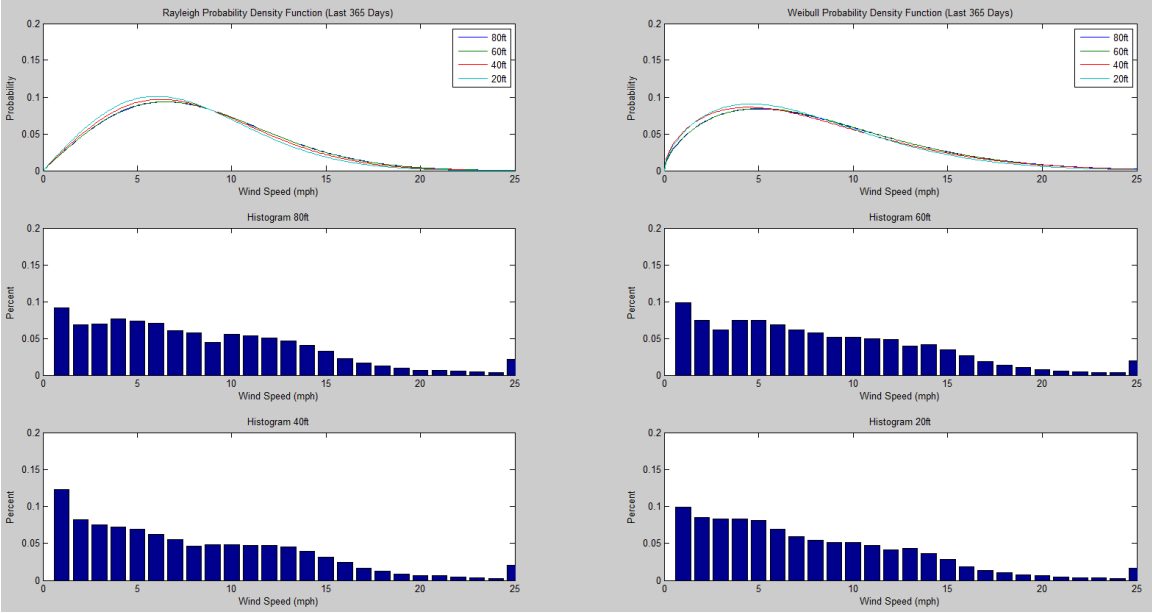


Figure 17: MATLAB plot of Rayleigh and Weibull distributions as well as histograms of wind speed for each of the 4 tower heights for the 365 days preceding June 5, 2011.

This plot shows that the data histograms at each tower height very closely approximate what is predicted by the Rayleigh and Weibull distributions. A Rayleigh distribution is given by:

$$p(U) = \frac{\pi}{2}\left(\frac{U}{\overline{U}^2}\right)\exp\left[-\frac{\pi}{4}\left(\frac{U}{\overline{U}}\right)^2\right] \qquad [21]$$

Where $p(U)$ is the probability of a wind speed U and $\bar{U}$ is the mean wind speed. A Weibull distribution is given by:

$$p(U) = \left(\frac{k}{c}\right)\left(\frac{U}{c}\right)^{k-1} \exp\left[-\left(\frac{U}{c}\right)^{k}\right] \quad [22]$$

Where k and c are given empirically by (Manwell et al. 2002):

$$k = \left(\frac{\sigma}{\overline{U}}\right)^{-1.086} \quad [23]$$

$$c = \overline{U}\left[0.568 + \left(\frac{0.433}{k}\right)^{(1/k)}\right] \quad [24]$$

Where σ is the standard deviation in wind speed.

Wind roses were created from the wind speed and direction data from the 60ft tower height. On an annual basis the wind predominately blows out of the west which can be seen in Figure 18. At the 60ft height the wind speed is at least 12mph roughly a third of the year and at least 8mph for roughly two thirds of the year when blowing out of the west. There is more variation in the direction and speed of the wind within each month of the year and when comparing different months as can be seen in Figures 19 and 20.

Many important observations about the wind distribution can be seen when the data is visualized in this way. First, the wind blows predominately out of the west in every month of the year except May and June. During each of those two months strong winds blow out of the northeast. Second, there is a distinct seasonal variation in the wind speeds. The winds are strongest during the spring (February through June) and weakest during the summer (July through September). This is encouraging because the winds are weakest during the summer months when campus is less populated and therefore

consumes much less energy. Last, the most variation in the directionality of the wind occurs during the winter (November through February) although there is a strong bimodality in the direction of the wind during June.

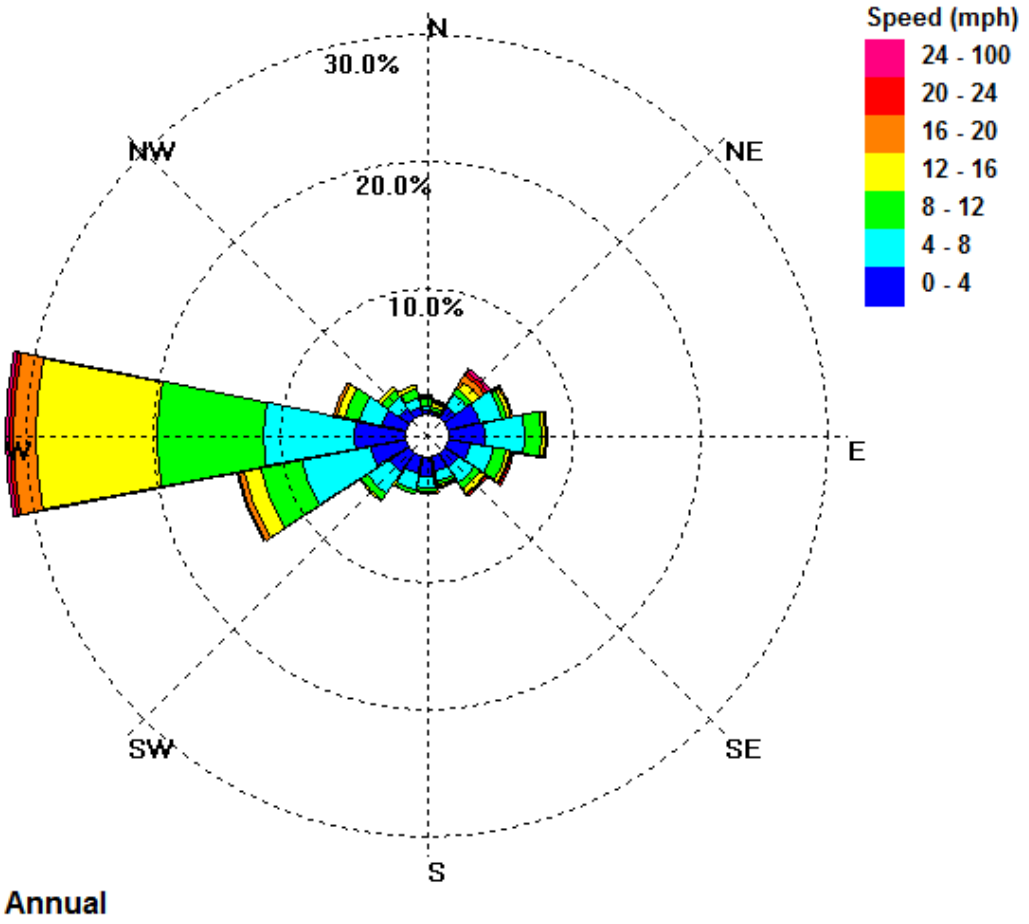


Figure 18: The annual wind speed and direction distribution from the data collected at the 60ft height on the MET tower for the year preceding June 5, 2011.

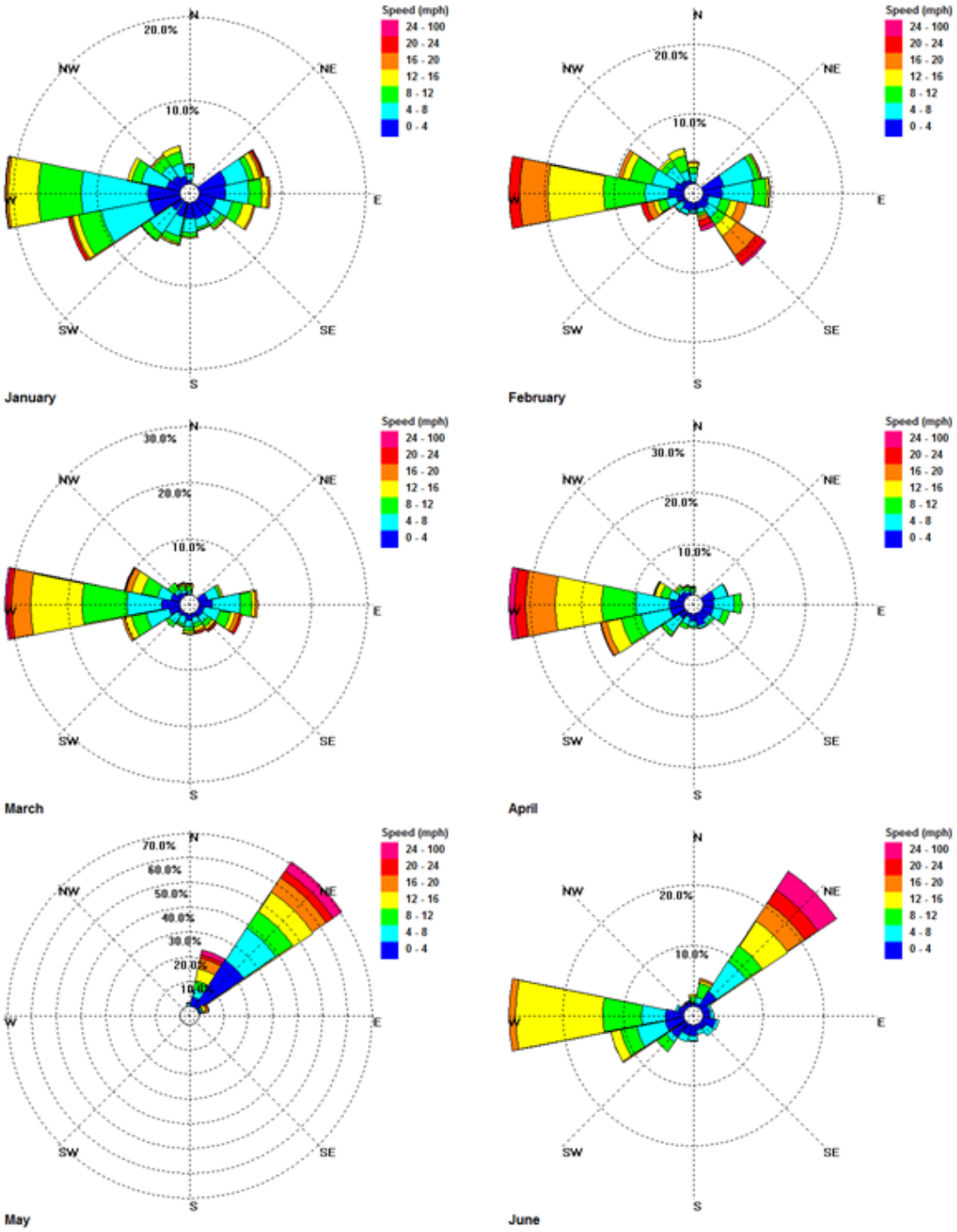


Figure 19: A month by month comparison of the wind speed and direction distribution for the months of January through June based on data collected from the 60ft height on the MET tower.

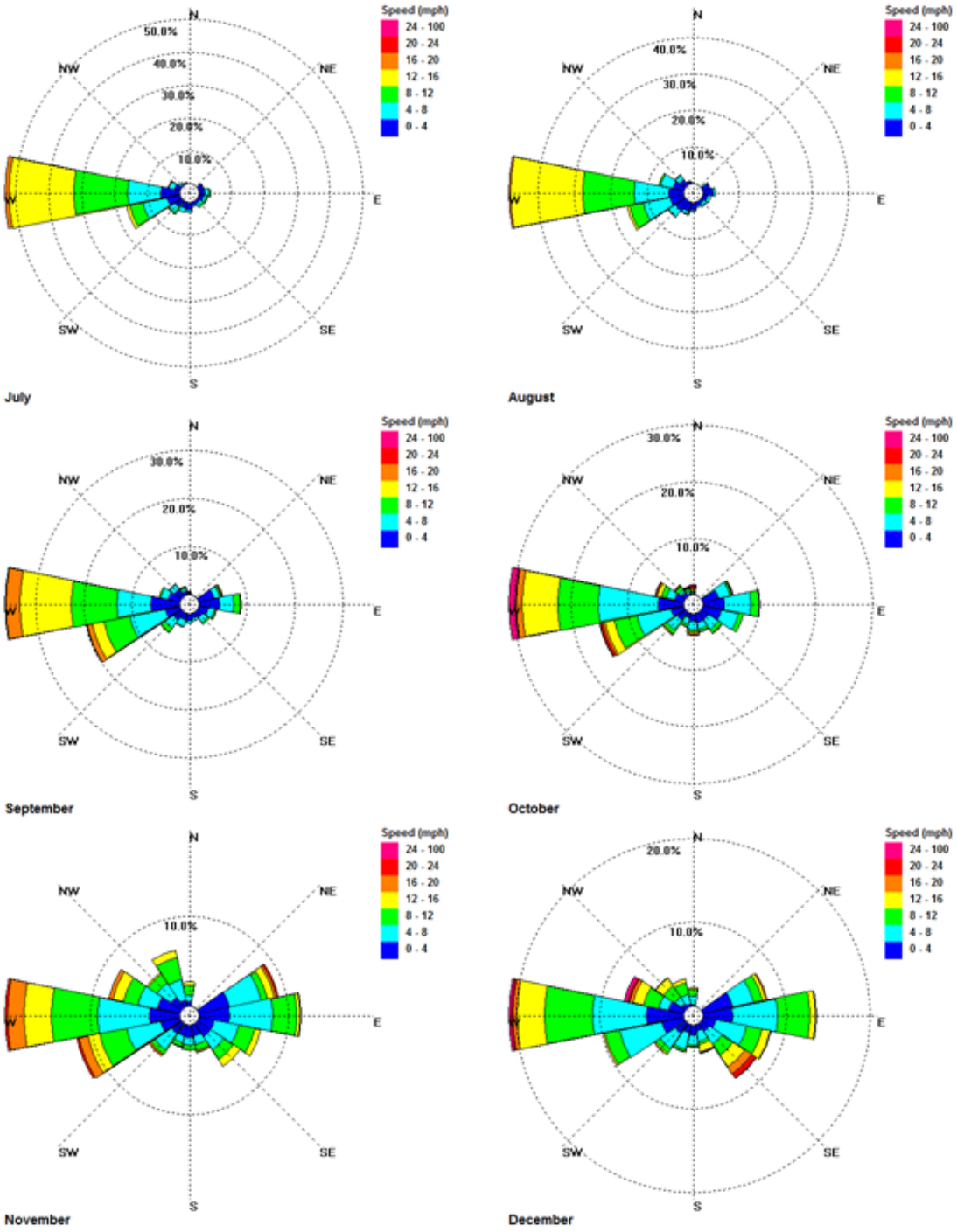


Figure 20: A month by month comparison of the wind speed and direction distribution for the months of July through December based on data collected from the 60ft height on the MET tower.

3.2 CFD Simulation Preprocessing

Before a CFD simulation can be run the geometry needs to be defined. It is important to model the geometry, specifically the topography, as accurately as possible

since the topography is the major factor in determining how the wind speed varies spatially. Then a mesh has to be generated to discretize the geometry. The mesh needs to have enough resolution to capture the flow variations accurately without being so fine that excess computing resources and time are wasted unnecessarily. Last, boundary conditions need to be defined which essentially provide a starting point for the simulation. These preprocessing steps are important and lay the groundwork for an accurate and meaningful solution.

3.2.1 Geometry Creation

The first step towards creating the geometry for the simulation is to model the topography. This was done by downloading digital elevation model (DEM) data from the ASTER-GDEM project (ASTER-GDEM 2009). The ASTER-GDEM project is a collaboration between the National Aeronautics and Space Administration (NASA) in the U.S. and the Ministry of Economy, Trade, and Industry (METI) of Japan. ASTER is the name of a satellite-born sensor which collects DEM data for all the land on earth hence GDEM which stands for global digital elevation model. ASTER-GDEM data has a spatial resolution of 30m in the horizontal plane and 20m in the vertical plane with 95% confidence. Data from the ASTER-GDEM project can only be downloaded in 1°x1° latitude/longitude tiles which is much too large of a data set for modeling Poly Canyon and Escuela Ranch. MICRODEM software was used to crop the DEM data down to what was needed for this project (Guth 2010). MICRODEM is a freeware microcomputer mapping program developed by professor Peter Guth at the U.S. Naval Academy. Figures 21 and 22 show the initial 1°x1° DEM data and the cropped DEM data respectively.

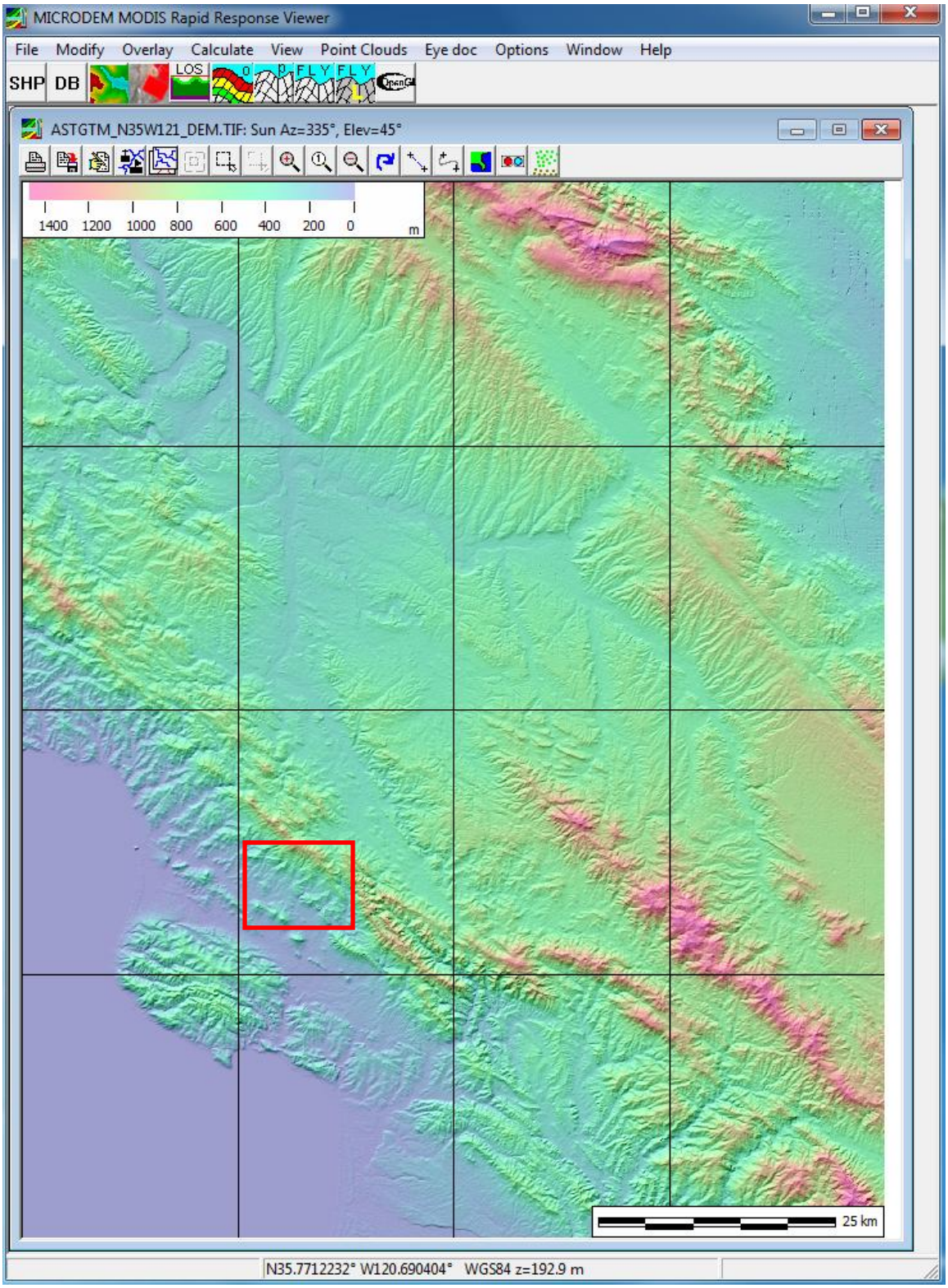


Figure 21: ASTER-GDEM 1°x1° DEM data for 35°N and 121°W. The red box indicates the cropped data.

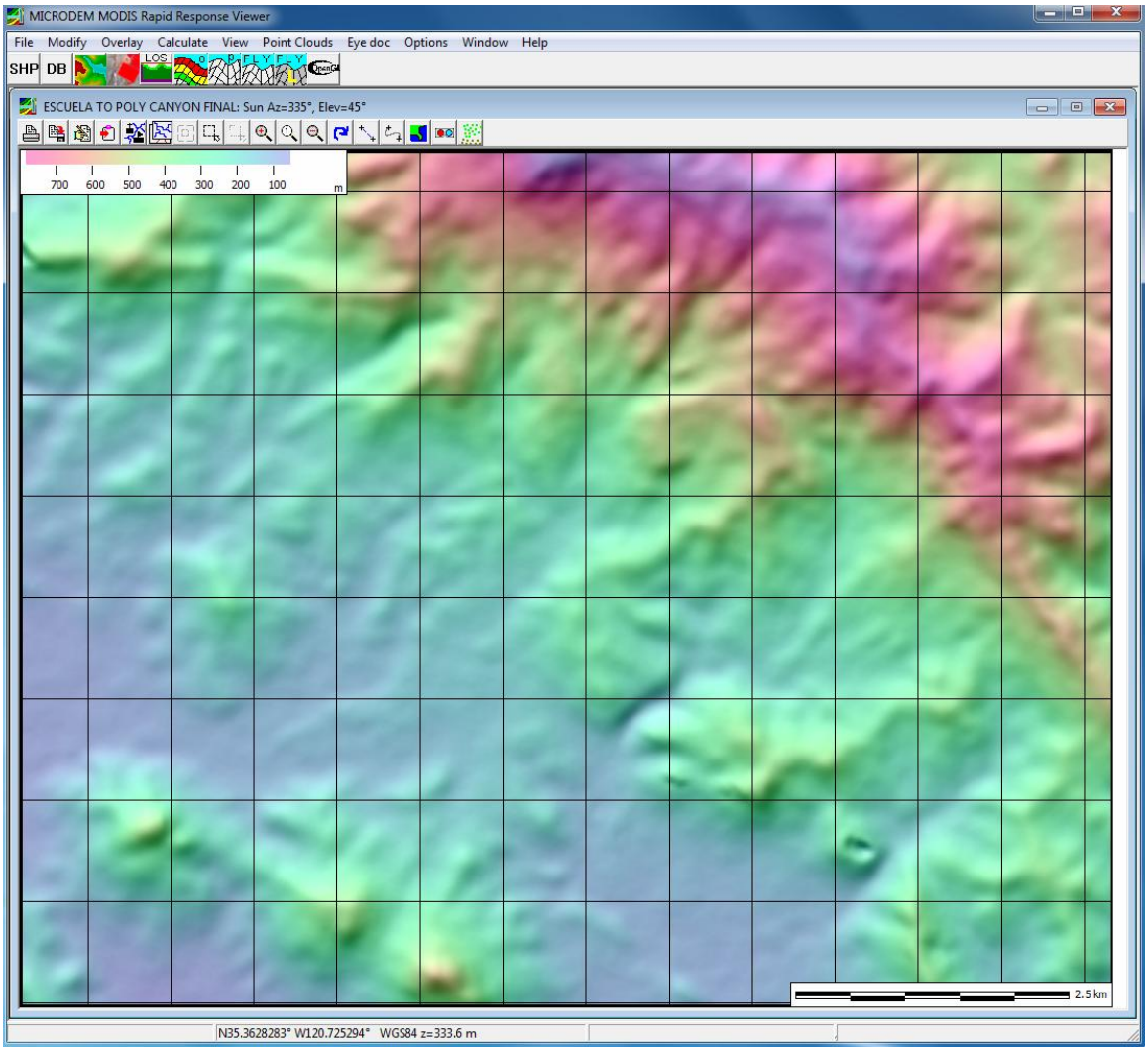


Figure 22: Cropped ASTER-GDEM data to encompass only the area of interest for wind resource assessment.

Once the data was reduced to only what was needed for the wind resource assessment it was exported as a set of x, y, and z data points in .XYZ format based on the Universal Transverse Mercator (UTM) grid. The UTM grid divides the globe into 60 zones along lines of longitude. The zones corresponding to the United States can be seen in Figure 23. Within each zone locations are denoted by a northing and easting measured in meters. The northing and easting values in a UTM grid do not correspond to true northing and easting values but are referenced perpendicular and parallel to the zone boundaries respectively. When the data points were projected on a traditional Cartesian

coordinate system they formed the shape of a parallelogram rather than a rectangle. This necessitated some manipulation of the data points.

Figure 23: UTM grid zones corresponding to the continental United States.

The data points were first manipulated using Excel because the elevation data was saved as the z component of the data during export from MICRODEM but the meshing software and FLUENT both use the y direction for elevation. In Excel the locations of the y and z columns of data were switched and the x column of data was multiplied by -1 to allow the data to conform to a right handed coordinate system. Once the re-indexing of the data was complete it was saved again in .XYZ format. Next, MATLAB was used to extract and retain only the data points lying within the rectangle defined by the four corners of the parallelogram lying within the parallelogram. An exaggerated graphical representation of this process is shown in Figure 24. The black box represents the true north, south, east, and west directions while the red box corresponds to the UTM grid.

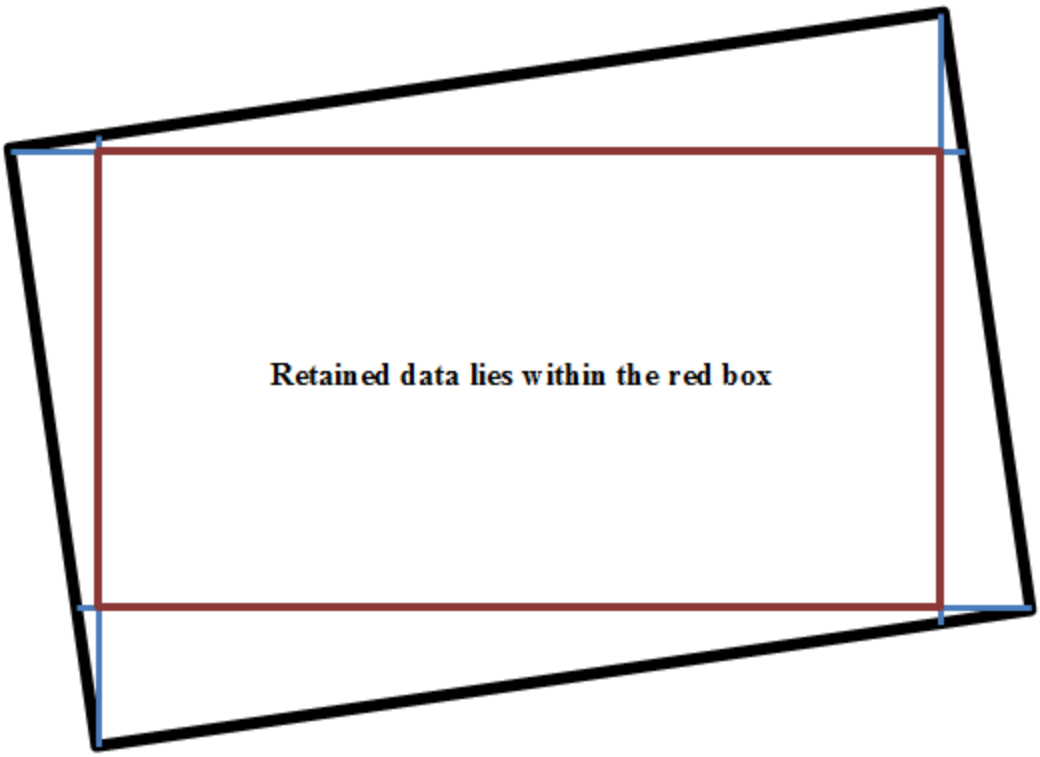


Figure 24: Graphical representation of the data retained by MATLAB from the UTM projection on a Cartesian coordinate system.

The re-indexed and retained data points were then opened in SolidWorks as a point cloud. The point cloud for this project included over 85,000 points and can be seen in Figure 25. The ScanTo3D add-in for SolidWorks was then used to create a surface from the point cloud. The surface created to model the topography for this project is shown in Figure 26. Then the surface was visually compared to Google Map's topology and was found to be a good representation of the area's topology. Last, the surface was saved as a .STL file for import into the meshing software.

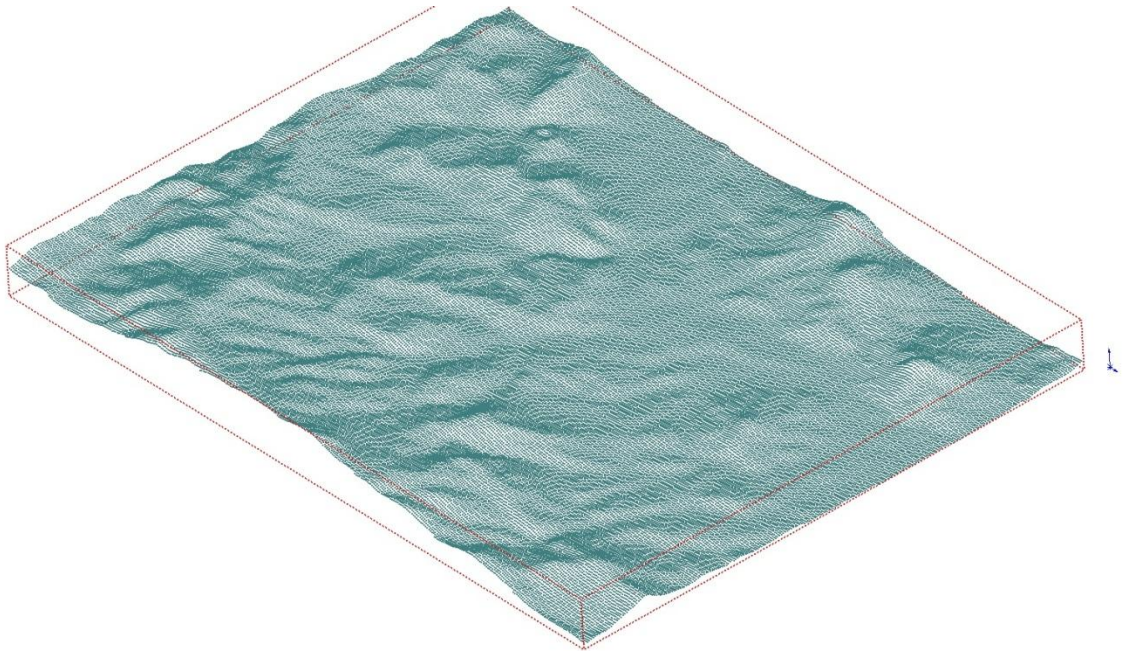


Figure 25: Point cloud in SolidWorks created from the re-indexed data points.

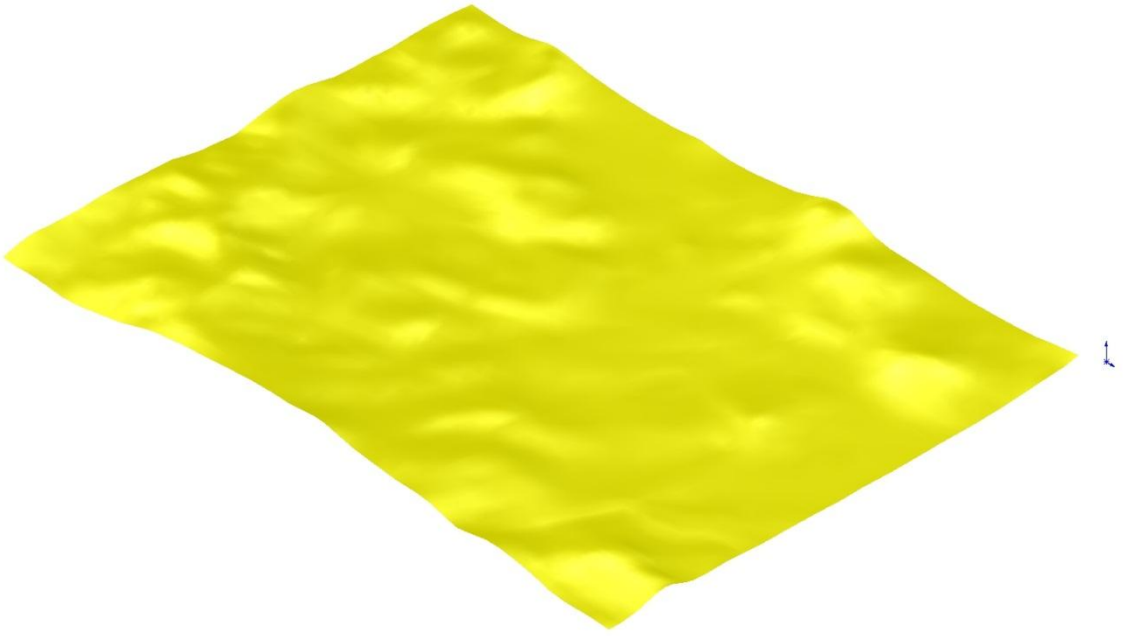


Figure 26: Surface created from the point cloud using the ScanTo3D add-in for SolidWorks.

3.2.2 Mesh Generation

The domain for the wind resource assessment is approximately 7.6 X 9.8km. To create a mesh with sufficient resolution while limiting the amount of computational

power needed, the domain is divided into two distinct zones. The first zone is the atmospheric boundary layer which is the region closest to the ground surface. The second zone is the far field. The far field is just above the atmospheric boundary layer and extends upward away from the ground. The height of each of the zones is determined by the results of the CFD simulations. The height of each of the zones is varied until the CFD solution no longer changes with varying zone height, at this point grid independence is reached.

Dividing the mesh into two zones allows for controlling the vertical resolution in each zone separately. This is important because finer vertical resolution is desirable in the boundary layer zone because velocity gradients in this zone are much greater than those in the far field zone. The atmospheric boundary layer zone is composed of the first few hundred meters above the ground surface. In this resource assessment the boundary layer zone was defined by a constant height above the ground surface rather than a constant elevation above sea level. This was done so that each column of cells within the boundary layer zone had the same height therefore, it allowed for the first cell height and inflation ratio to remain constant throughout. Maintaining the height of the first cell above the ground surface as well as the inflation ratio over the whole area of interest is important for accurate CFD results.

The far field zone begins immediately above the boundary layer zone and extends up to a constant elevation which is determined by grid independence. This means that each column of cells in the far field has a different height. Although the height of each column of cells varies in the far field, the number of cell layers remains constant causing the inflation ratio to vary from column to column. This is acceptable outside the

boundary layer because velocity gradients are much smaller in the far field than in the boundary layer. In addition, the vertical resolution in the far field is much coarser than in the boundary layer zone to save on computing power required to run the CFD simulation. Figure 27 below shows a small cross section of the mesh which illustrates the difference in vertical resolution between the two zones as well as the effects of having the boundary layer defined by a constant height above the ground surface while the far field is defined by a constant elevation. The red line indicates where the boundary layer zone ends and the far field zone begins.

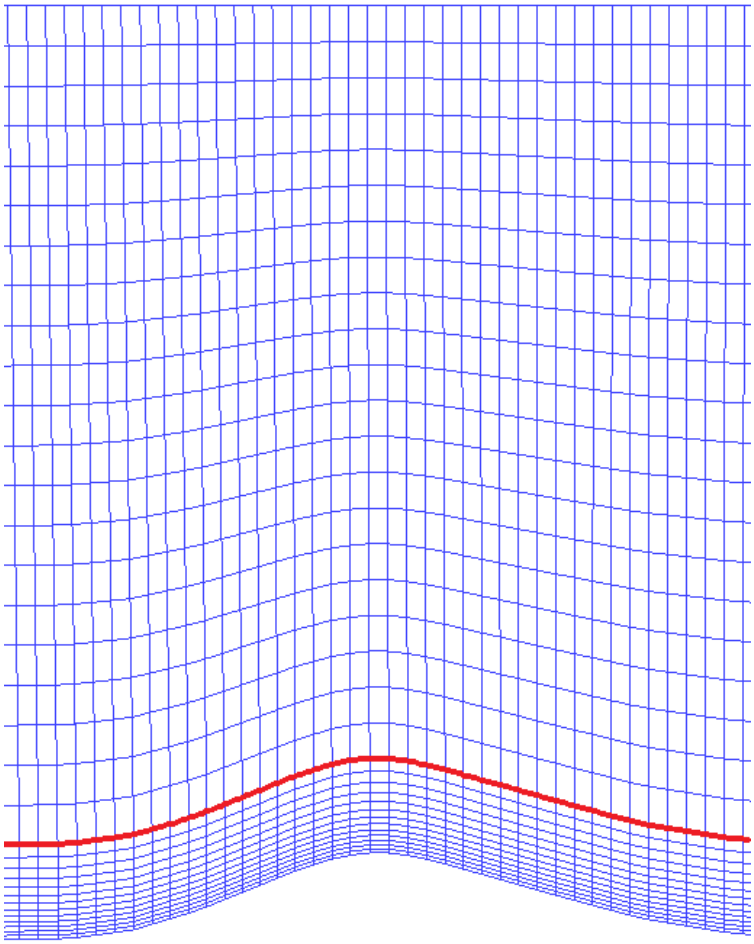


Figure 27: Small cross section of the mesh illustrating the boundary layer and far field zones.

3.3 FLUENT CFD Simulation Settings

Before running a simulation in FLUENT many different parameters, boundary conditions, and models need to be defined. For the Cal Poly wind resource assessment FLUENT was set up using double precision for all calculations to reduce rounding errors. The solver was set for steady, pressure-based calculations with an absolute velocity formulation. Gravity was set to $9.81 m/s^2$ in the negative y direction. To model turbulence the realizable K-ε model was used which is a slightly modified version of the standard K-ε model described above. The constants used in the turbulence model were not modified and standard wall functions were used.

Next, the boundary conditions were defined. The simulation domain is shown in red in Figure 28 below. The wind direction data collected from the MET tower, depicted as a blue X in Figure 28, indicates that the wind predominately blows out of the west so all simulations were run with the western edge of the domain as the inlet, the eastern edge as the outlet, and the north and south edges as the sides surfaces. The simulation domain is so much larger than the area of interest for the wind farm so that the MET tower is located on the boundary of the domain so the MET data can be used to create the inlet boundary condition. The boundaries of the land owned by Cal Poly are shown in black.

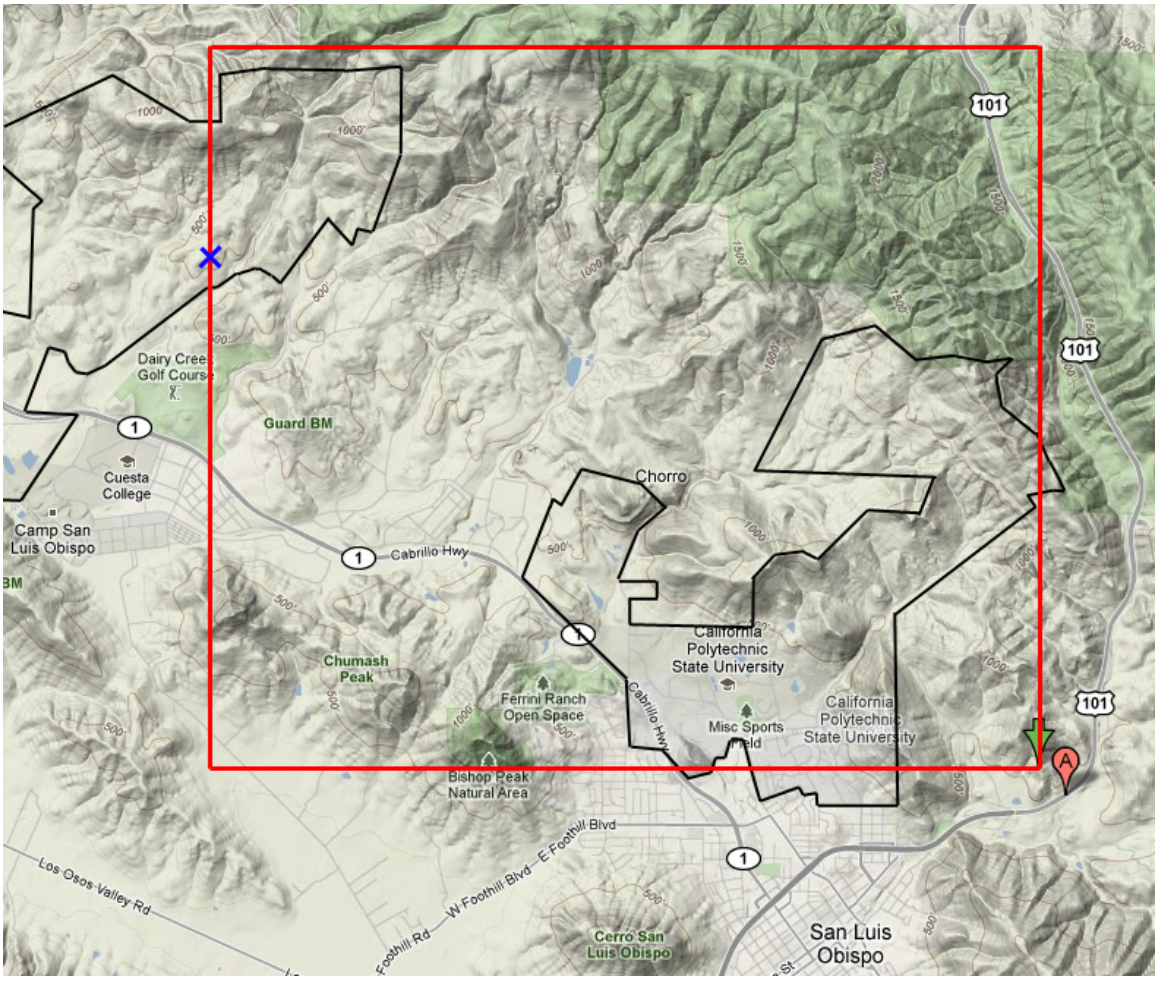


Figure 28: Google maps image showing the extent of the simulation domain and location of the MET tower.

The inlet surface was set as a velocity-inlet with the magnitude described by a user defined function which can be found in Appendix D. The user defined function is a simple power law velocity profile with a shear exponent of 1/7 and reference height and velocity based on the collected data from the MET tower. The power law velocity profile was used along the entire inlet plane even though the data collected at the MET tower is only representative of one discreet location along that boundary. The outlet surface was set as a pressure outlet at atmospheric pressure. The inlet and outlet turbulence parameters were defined by a turbulence intensity of 10% and a turbulent viscosity ratio of 10. The top and side surfaces were defined as symmetry surfaces which means that

gradients across those surfaces are all set to zero. Last, the ground surface was defined as a wall obeying the no slip condition.

Then all residuals were set to 1e-6 and the solution was allowed to run to convergence with the SIMPLE pressure-velocity coupling scheme and spatial discretization for gradients using a least squares cell based scheme, pressure using the standard scheme, and momentum and turbulence using first order upwinding. Then after convergence the spatial discretization for pressure was changed to second order and the discretization for momentum and turbulence were changed to second order upwinding and the solution was run to convergence once again.

Chapter 4: Grid Independence, Simulation Validation, and Results

Solutions need to be evaluated for grid independence by varying the mesh resolution and extent while maintaining the simulation settings. If grid independence isn't established the solution resulting from the simulation is invalid and inaccurate. Grid independence ensures that for a given simulation set up, the solution is as accurate as possible and does not depend on the geometry of the mesh. Finally the simulation settings need to be validated. Even if grid independence is achieved there is no guarantee that the simulation settings are appropriately defined which is why a solution still cannot be relied upon until the simulation settings are validated. This is most often done by replicating the results of other CFD simulations or by comparing to collected data for similar geometry and flow conditions.

4.1 Establishing Grid Independence

With the simulation settings described above solutions are obtained for meshes with varying resolution. In addition, the vertical extent of both the boundary layer and far field are varied. Grid independence studies are done first by varying the resolution horizontally, then vertically. The solutions from each of these mesh configurations are compared to one another on a plane of interest. In this case the plane of interest is 80m above the ground surface which corresponds to the typical hub height for a commercial, utility grade wind turbine.

4.1.1 Mesh Resolution

The first step in establishing grid independence is to investigate the effects of varying mesh resolution on the simulation results. For a mesh with a boundary layer zone height of 250m and a far field zone which extends to 1500m above the lowest point on

the topography, simulations were run for horizontal mesh resolutions of 100m, 50m, and 25m on a plane parallel to the ground. At this point the height of the boundary layer zone and extent of the far field zone are just educated guesses and must be investigated as well. The hub height velocity distributions for the simulations with 100m and 50m resolutions are shown in Figure 29 on the left and right respectively. There are clear visually detectable differences between the simulations with 100m and 50m resolutions so further refinement of the mesh is necessary.

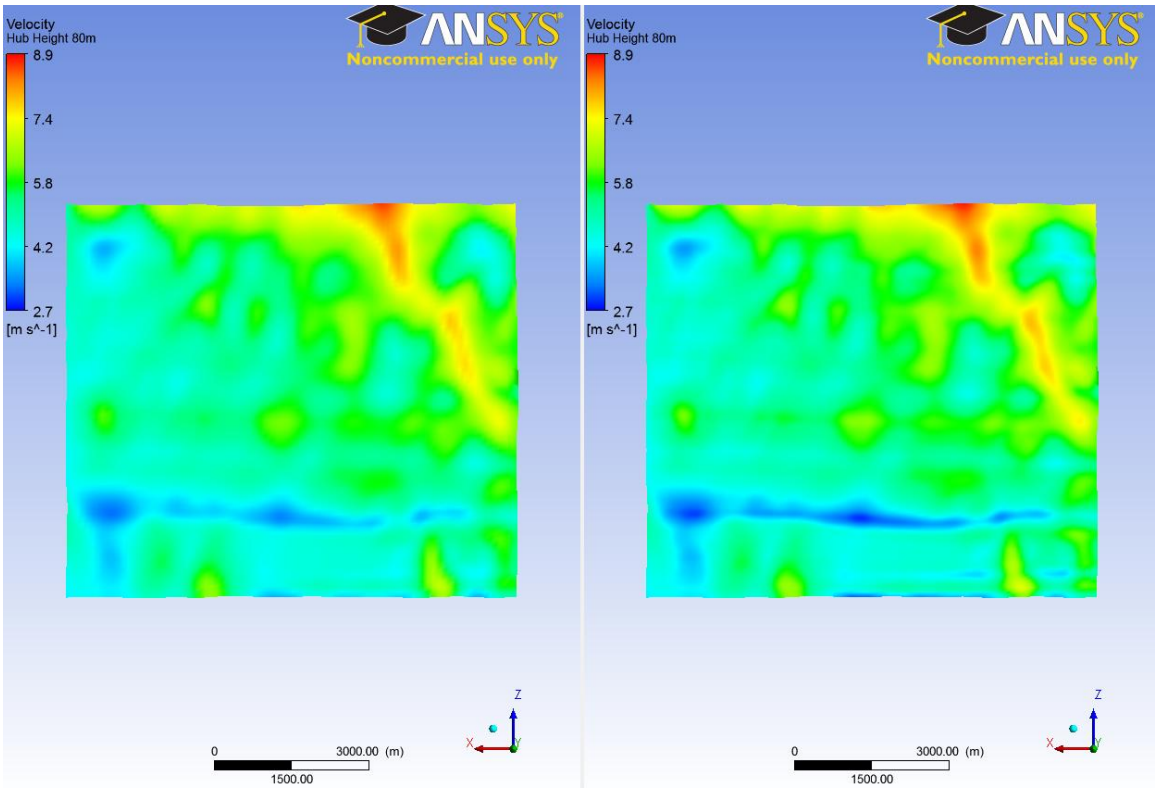


Figure 29: Left: velocity distribution for a mesh resolution of 100m. Right: velocity distribution with a mesh resolution of 50m.

Figure 30 shows the simulation results for meshes with 50m and 25m on the left and right respectively. In this case the differences in the velocity distributions are much more subtle. A comparison of the two which is shown in Figure 31 illustrates that the differences between the velocity distributions are negligible when refining the mesh

resolution from 50m to 25m. Therefore a mesh with 50m spatial resolution will be used in all future simulations because it requires less computational power than a 25m grid.

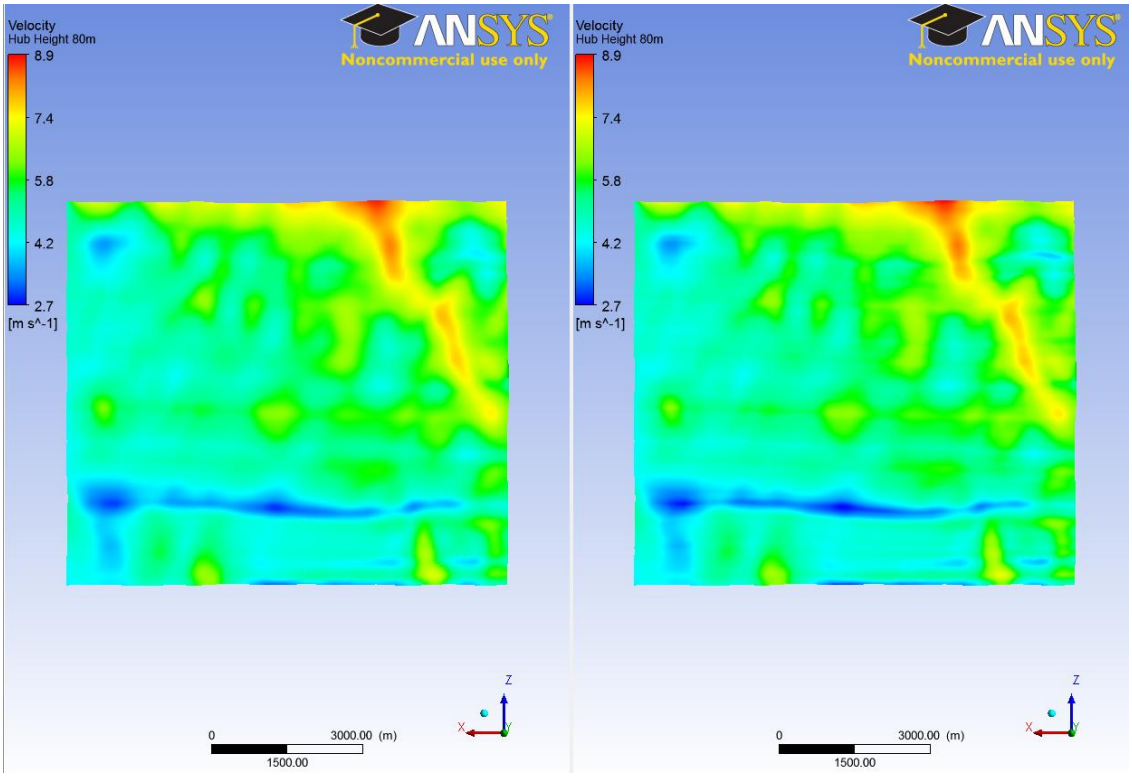


Figure 30: Left: velocity distribution for a mesh resolution of 50m. Right: velocity distribution with a mesh resolution of 25m.

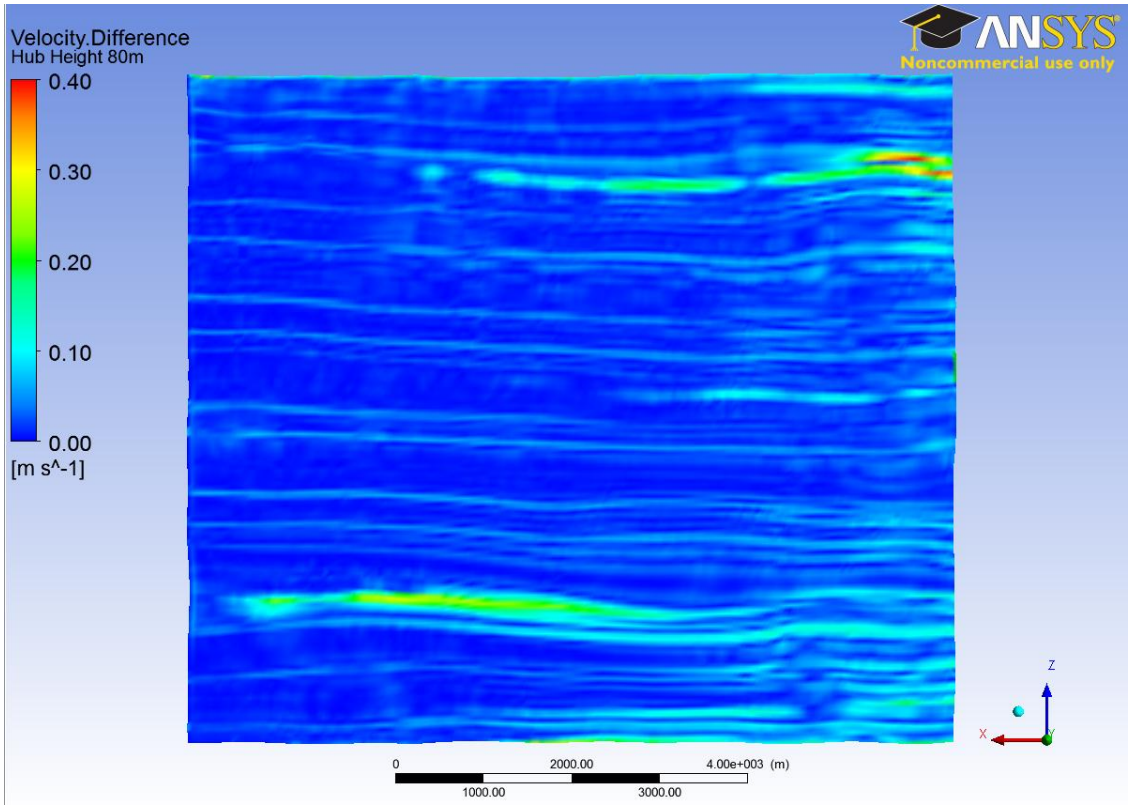


Figure 31: Difference in velocity between simulations with 50m and 25m resolutions.

4.1.2 Far Field Zone Height

After establishing an appropriate mesh resolution it is important to investigate the effect of varying the height of the far field zone. Although the velocity gradients within the far field zone are smaller than inside the boundary layer zone, the extent of the far field zone has a large effect on the simulation results due to the symmetry boundary conditions imposed on the top and side surfaces. Simulations were run for four different far field heights 1500m, 2000m, 2500m, and 3000m above the lowest point of the topography. In each of these cases the boundary layer zone is 250m tall which must be investigated once the proper extent of the far field is determined. A spatial resolution of 50m is used for each of these four cases as determined above. The vertical resolution of the cells in the far field zone is 50m and more cells were added to increase the height of the far field zone. The results of the simulations with a far field of 1500m and 2000m can

be seen in Figure 32 on the left and right respectively. Figure 33 shows the results of the 2000m and 2500m simulations on the left and right respectively and Figure 34 compares the 2500m and 3000m far field simulations.

There are visually detectable differences in the hub height velocity distributions until the height of the far field reaches 2500m. At that point it is not clear whether an increase to 3000m has an effect on the solution. These two velocity distributions are compared and the results are visible in Figure 35.

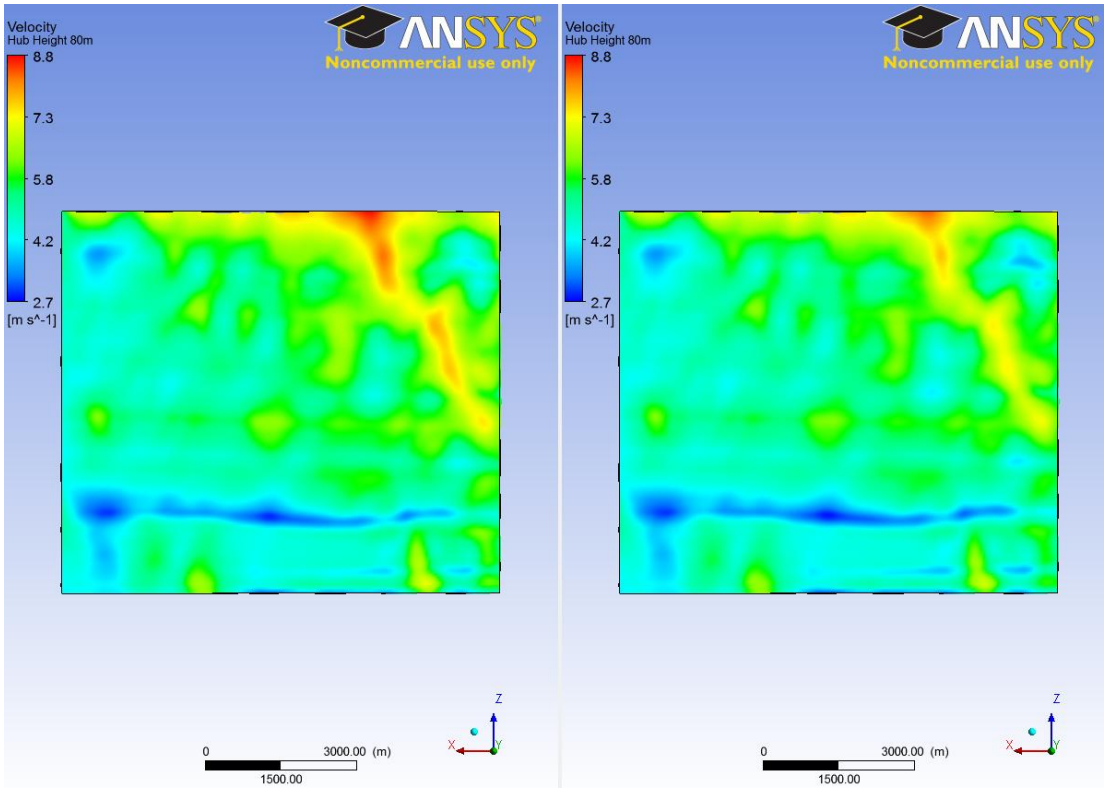


Figure 32: Left: velocity distribution for a 1500m far field. Right: velocity distribution for a 2000m far field.

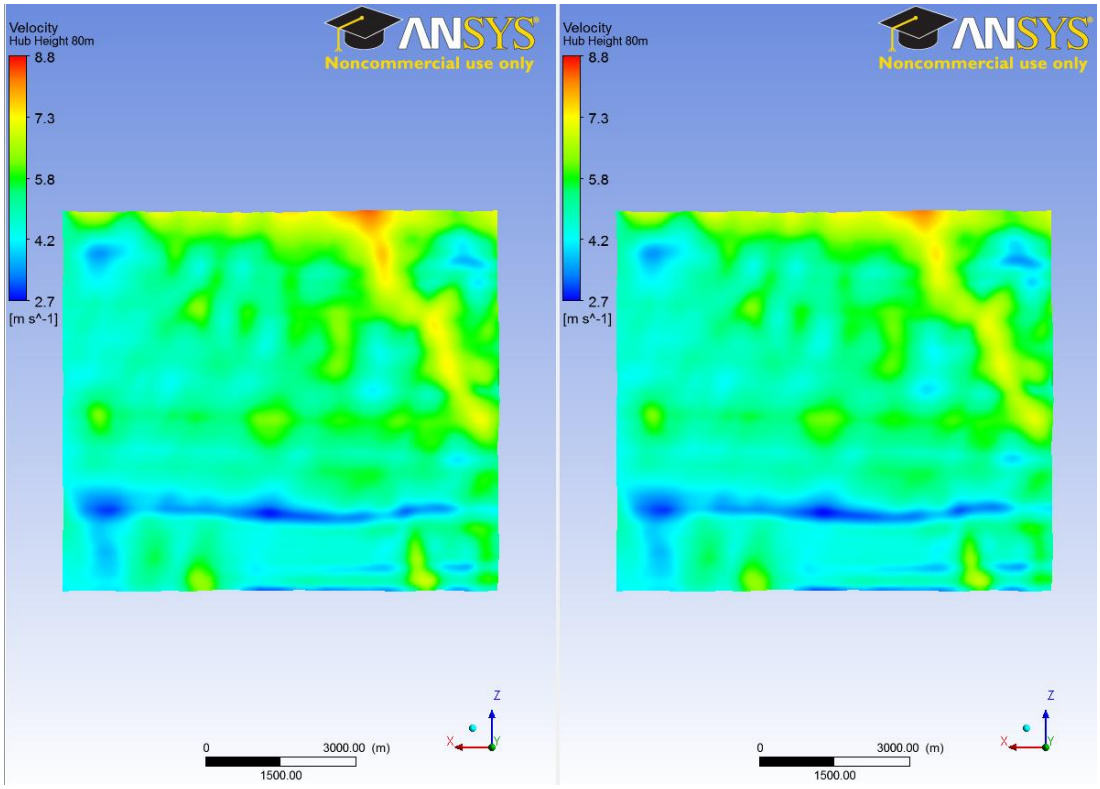


Figure 33: Left: velocity distribution for a 2000m far field. Right: velocity distribution for a 2500m far field.

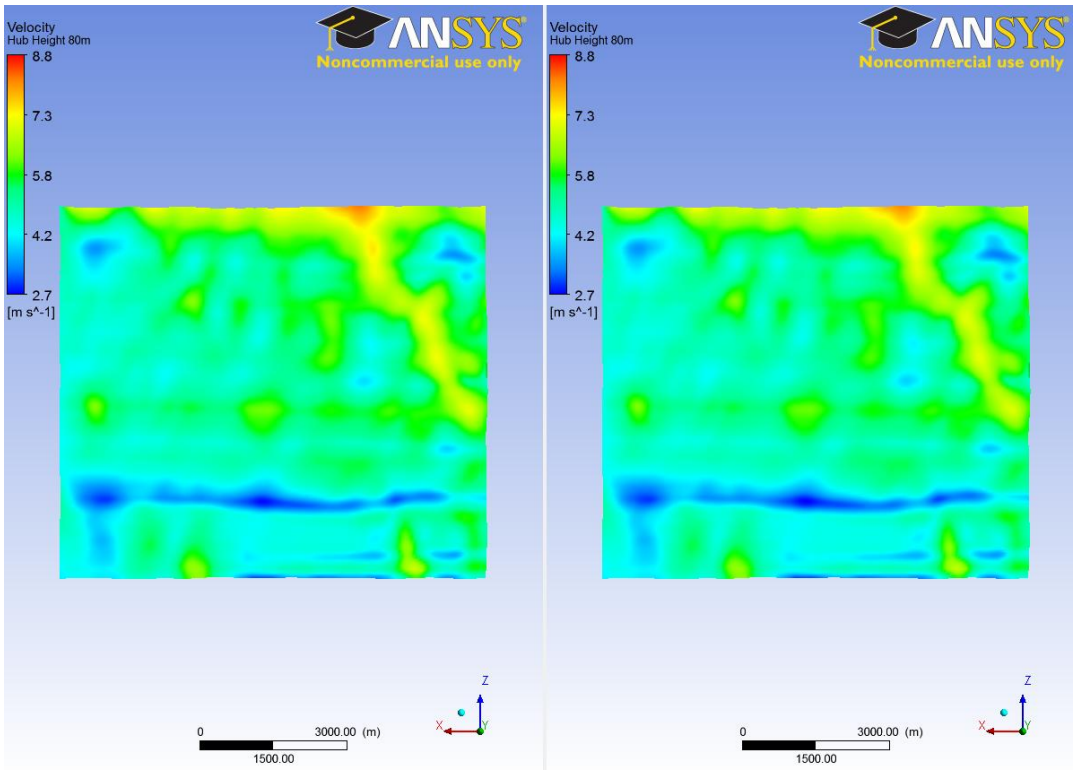


Figure 34: Left: velocity distribution for a 2500m far field. Right: velocity distribution for a 3000m far field.

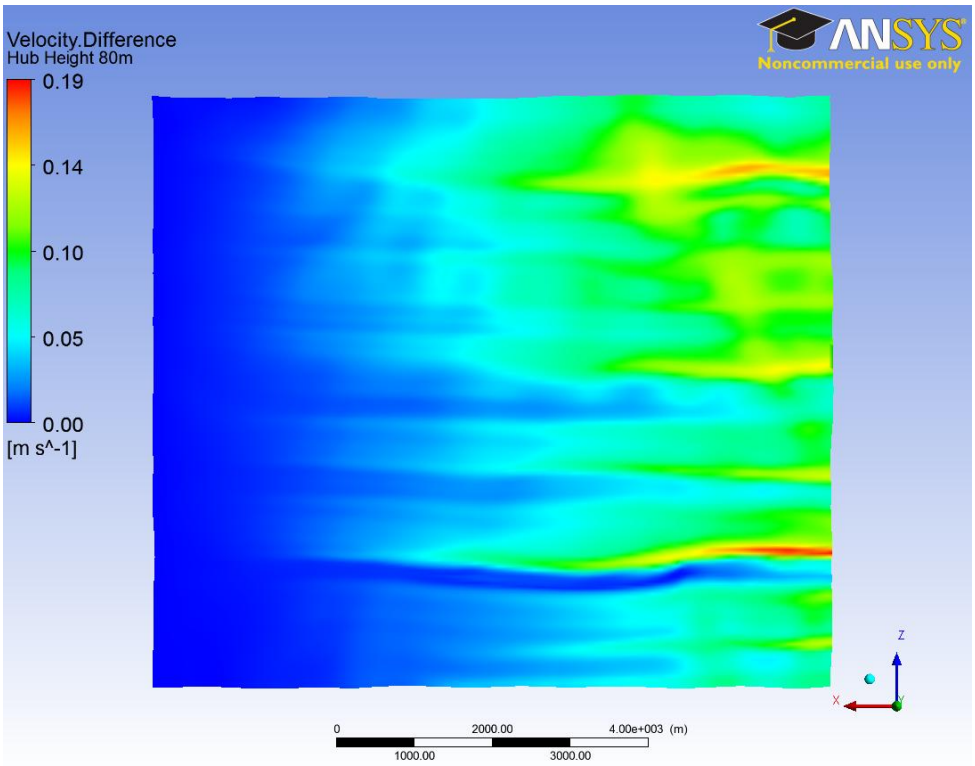


Figure 35: Difference in velocity at hub height between the 2500m and 3000m far field zone height simulations.

There are distinct differences between the two velocity distributions however they are acceptably small. The differences between these two cases are only approximately 6% at worst. The largest difference between the two cases is 0.19m/s and occurs where the velocity is approximately 3m/s which equates to a difference of just over 6%. The highest differences occur where the velocity is lowest; the difference in velocity over the majority of the domain is well under 6% and a far field zone height of 2500m is acceptable and will be used in all future simulations with a spatial resolution of 50m.

4.1.3 Boundary Layer Zone Height

Last, the height of the boundary layer zone was investigated. Two separate simulations were run, one with a 250m tall boundary layer zone and the other with a 500m tall boundary layer zone. In both cases the vertical limit on the far field zone was 2500m above the lowest point on the topography and the horizontal spatial resolution was

50m. The simulation results showing hub height velocity can be seen in Figure 36 with the 250m zone on the left and the 500m boundary layer zone on the right. In each case the height of the cell closest to the ground was 10m and an inflation ratio of 1.10 was used throughout the boundary layer zone.

Visually it appears that the results of these two simulations are identical but using a comparison tool in ANSYS Workbench it becomes clear that they are not. The results of the comparison are shown in Figure 37 below. Although the simulation results are not identical careful examination of the scale in Figure 37 shows that the differences are negligibly small and the absolute value of the largest differences are in locations where the velocity is highest. This indicates that a boundary layer zone only 250m tall is grid independent. The 250m boundary layer zone height is used in all future simulations because it requires less computational power than a 500m tall zone.

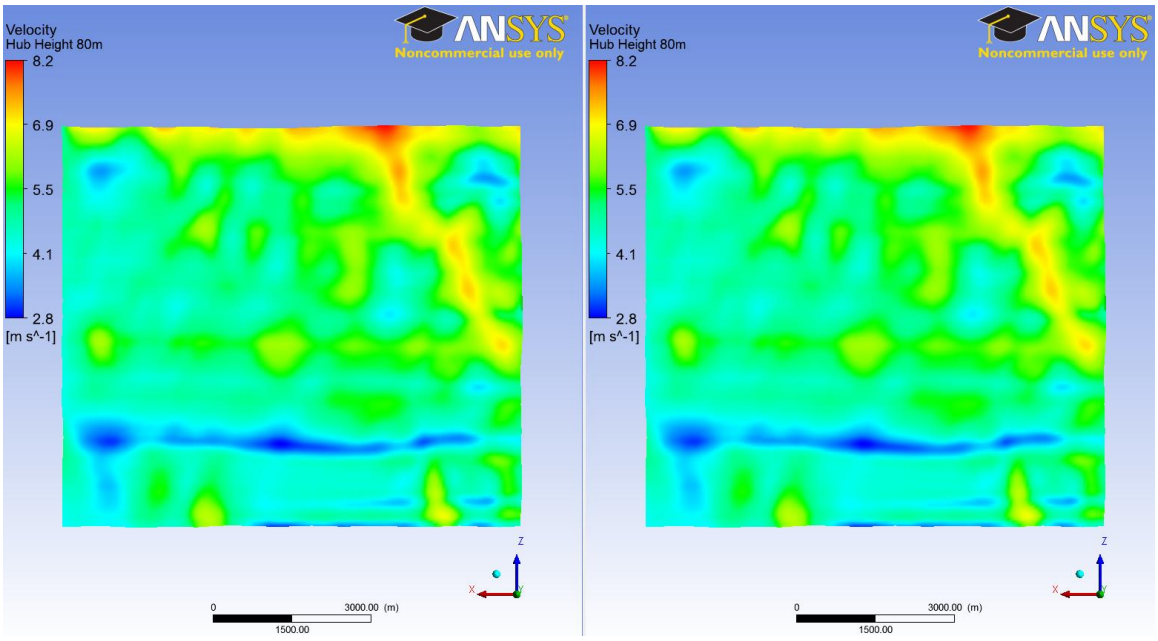


Figure 36: Simulation results investigating the effect of varying boundary layer zone height.

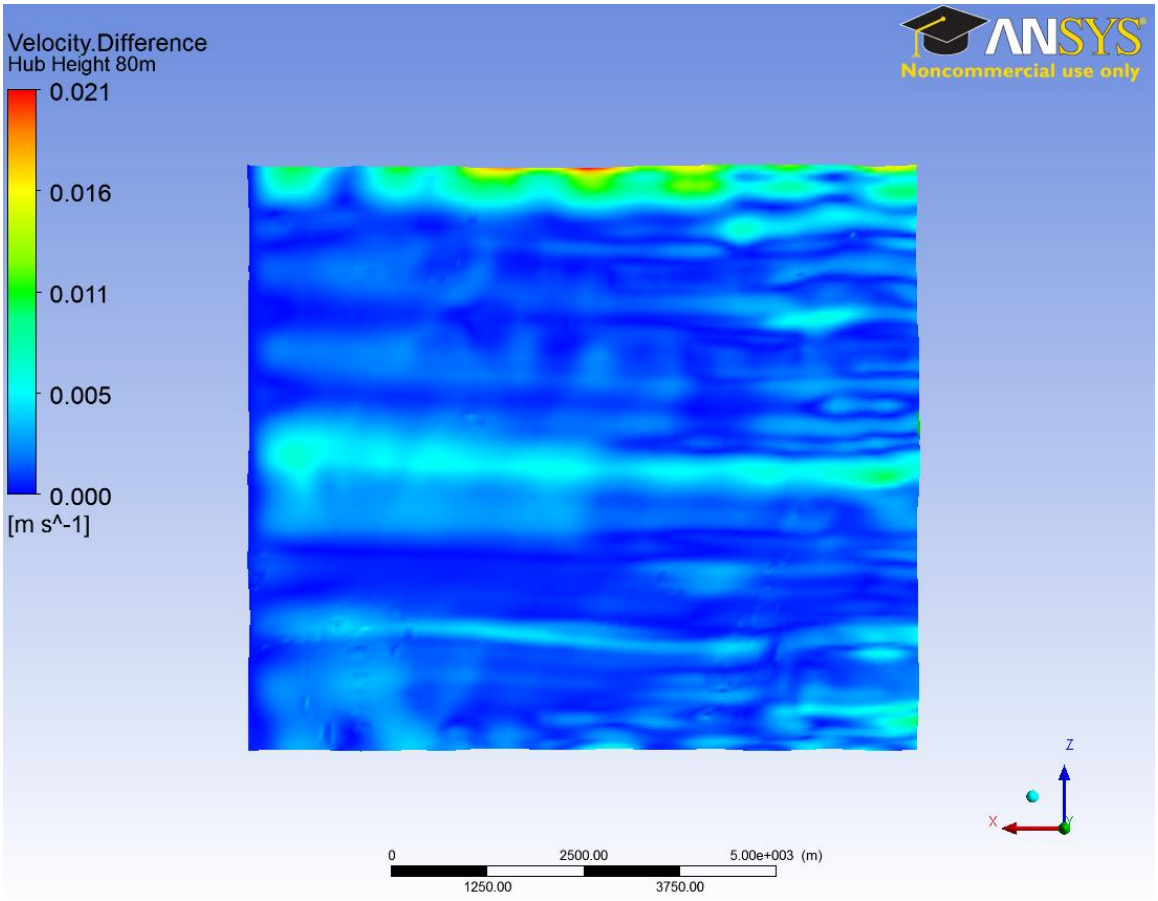


Figure 37: Difference in velocity at hub height between the 250m and 500m boundary layer zone simulations.

This completes the grid independence study of the simulation results. The ideal grid for these particular simulation settings and specific topography has a 50m spatial resolution, the far field zone extends to 2500m above the lowest point on the topography, and incorporates a 250m tall boundary layer zone within which the mesh is refined towards the ground surface.

4.2 Validating the Simulation Settings

An extensive study of the effect of hills on the atmospheric boundary layer was carried out in the early 1980's at Askervein Hill on the island of South Uist in Scotland. The objective of the study was to better understand atmospheric boundary layer flow over hills for the purpose of siting wind energy conversion systems. The atmospheric

boundary layer flow was investigated by instrumenting the hill with multiple MET towers each collecting data using one or more anemometers at various heights. Recreating the results of this study using CFD simulations has been done many times (Eidsvik, K.J. 2005). In addition, the results of this study have been used to validate CFD simulations for other locations (Russell 2009).

The hill is a little over 100m tall at the tallest point and is essentially elliptical in shape with a 1km long minor axis and a 2km long major axis. The hills major axis runs approximately from NW to SE. The prevailing wind during the time of the study blew mostly out of the southwest where the terrain is very flat extending to the ocean approximately 3km away. A reference site (RS) was constructed 3km to the SSW of the hill to measure the undisturbed upstream flow. MET towers were erected in a line (line B) roughly along the hills major axis which runs through the hills tallest point (HT) and center point (CP). Towers were also installed in a line (line A) perpendicular to the major axis running through HT as well as in a line (line AA) which is parallel to line A and runs through CP. The topography of the hill, the surrounding area, and the locations of the MET towers on the hill are shown in Figures 38 and 39.

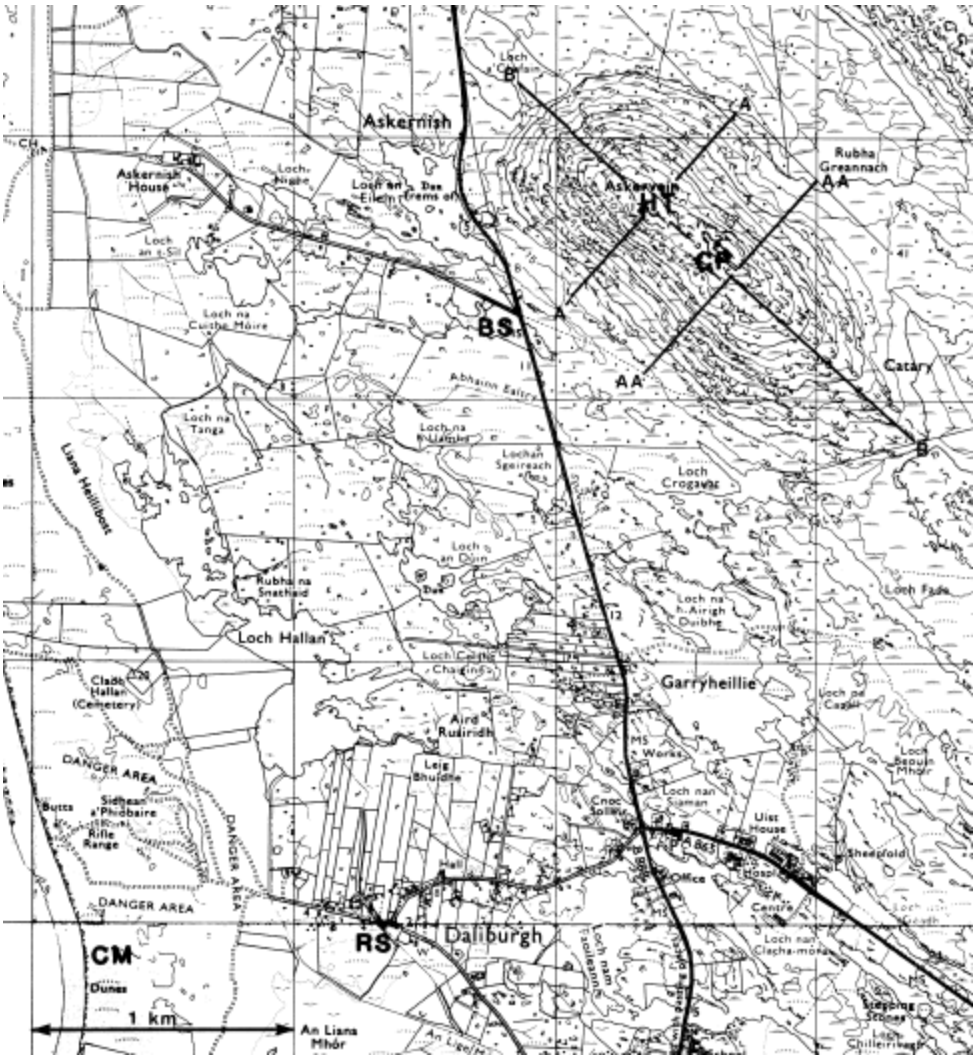


Figure 38: Map showing the relation of the hill to the reference site.

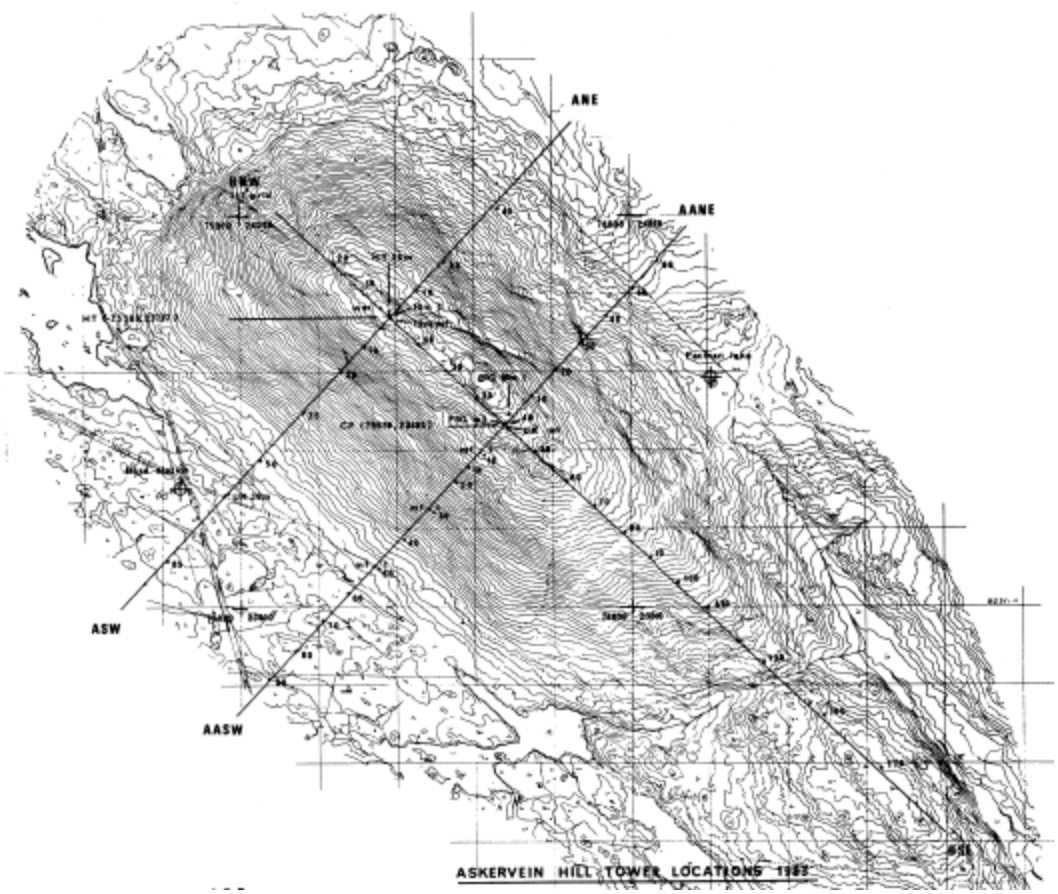


Figure 39: Topography of Askervein hill and locations of the MET towers.

To validate the settings for the Cal Poly CFD simulation, the same settings were applied to a CFD simulation of Askervein hill. The results of the Askervein hill simulation were then compared to measured data from the study in 1983. In the simulation of Askervein hill the wind was set to blow out of the south to match a particular period of collected data from the study. A power law velocity profile based on data collected at RS was used for the inlet. The reference height used was 24m and the wind speed was 10.43m/s. A shear exponent of 1/7 was used to match the Cal Poly simulation settings although a more accurate shear exponent could be calculated from data at various heights on the tower at RS.

Once the Askervein hill simulation was set up and grid independence was established the resulting velocity distribution could be compared to the data collected during the study. The majority of MET towers used in the Askervein hill study only measured the wind speed at 10m although select towers had multiple measurement heights. Figure 40 shows the velocity distribution 10m above the ground to give a general idea of the wind speed measured by the majority of anemometers during this particular period of the study.

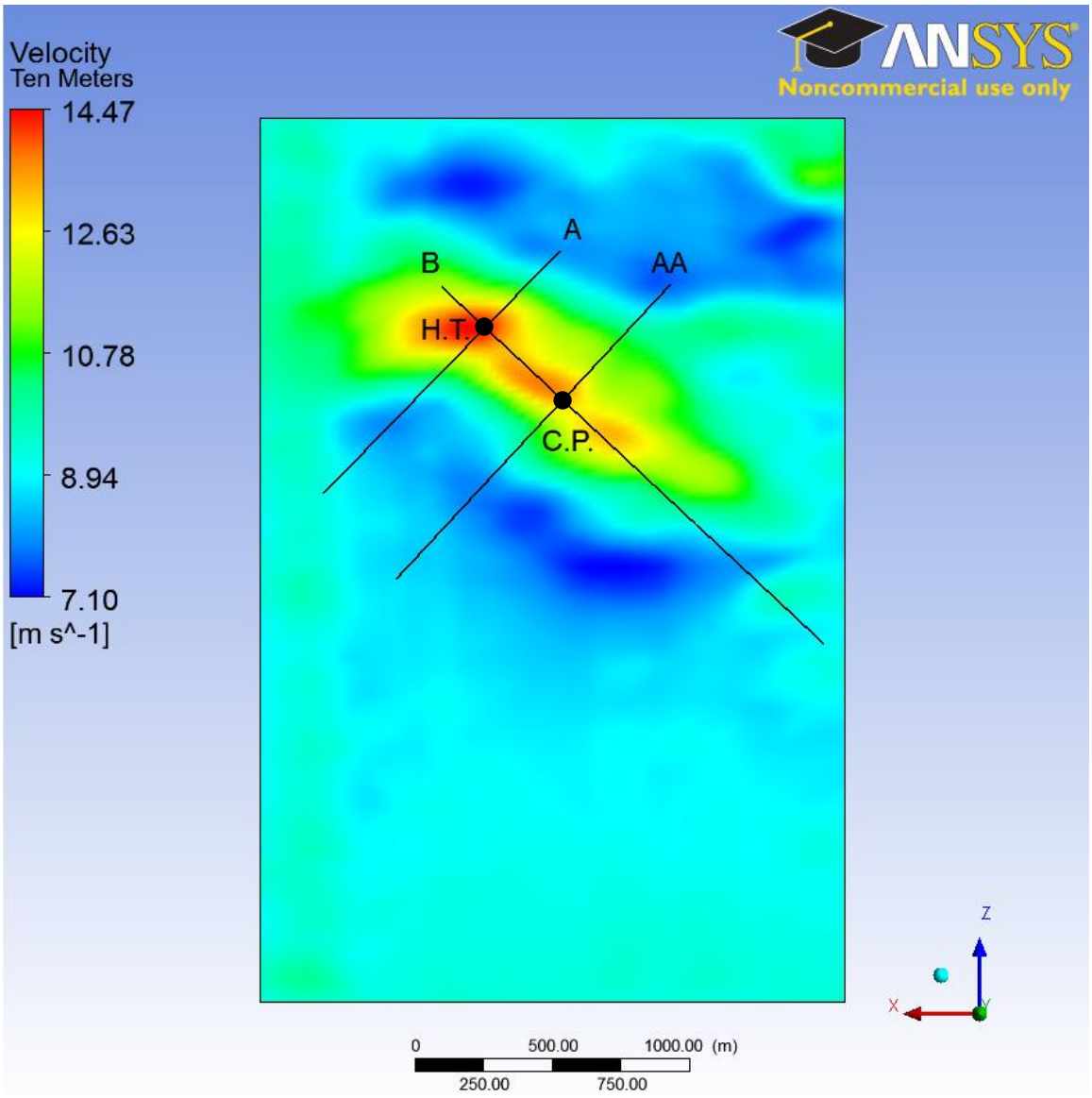


Figure 40: Velocity distribution 10m above the topography of Askervein hill.

MET tower locations were designated with abbreviations consisting of three parts. The first part signifies which line the tower is on (A, AA, or B). The second part designates the direction the tower lies from HT on lines A or B, or the direction from CP on line AA. The last part of the tower abbreviation is a means of describing the approximate distance from HT or CP in tens of meters. For example tower AASW60 stands for a tower on line AA which is SW of CP approximately 600m. Tables 3 and 4 compare the results of the Askervein hill simulation to the collected data from the study.

Table 3: Comparison of simulation results to measured data for two towers with multiple measurement heights.

Tower	Measurement Height (m)	Measured Wind Speed (m/s)	Calculated Wind Speed (m/s)	Error (%)
HT	3.9	14.13	11.08	-21.6
HT	5	15.20	12.51	-17.7
HT	8	15.51	14.45	-6.8
HT	15	15.79	14.89	-5.7
HT	24	15.62	15.21	-2.6
HT	34	15.19	15.40	1.4
HT	49	15.45	15.51	0.4
ASW60	6	8.71	7.31	-16.1
ASW60	20	10.29	9.58	-6.9
ASW60	31	10.60	10.55	-0.5

The highlighted row in Table 3 signifies a note from the study which indicates that the measured value is suspected to be inaccurate.

Table 4: Comparison of simulation results to data measured at 10m on various MET towers.

Tower	Measured Wind Speed (m/s)	Calculated Wind Speed (m/s)	Error (%)
ASW85	9.5	9.44	-0.6
ASW50	8.7	8.11	-6.8
ASW35	8.8	8.53	-3.1
ASW20	11.4	11.99	5.2
ASW10	13.4	12.98	-3.1
ANE10	12.5	12.69	1.5
ANE20	7.1	7.52	5.9
ANE40	7.4	7.71	4.2
CP	13.24	13.27	0.2
AASW10	12.73	11.41	-10.4
AASW20	10.97	10.06	-8.3
AASW30	9.86	9.06	-8.1
AASW40	8.49	8.05	-5.2
AASW50	7.96	7.95	-0.1

The comparison between the simulation results and collected data suggest that the simulation settings are well suited for this topography. Out of 14 MET towers examined which take measurements at 10m, the calculated wind speed of all of them are within

approximately ±10% of the measured wind speeds. In addition of the two towers which have multiple measurement heights, the measured and calculated wind speeds are within ±7% for measurement heights above 6m.

Although the results of the Askervein hill simulation are favorable there are some important things to note. The error between measured and calculated wind speeds at measurement heights at or below 6m were much worse than at heights above 6m; as high as 21.6%. This suggests that the simulation settings produce results which are likely not accurate near the ground surface because surface roughness is not properly taken into account. Also the height of the cells closest to the ground in the simulation is 10m which likely causes the greater error at measurement heights below 10m. Also it should be noted that tabulated data for the MET towers on line B and locations NE of CP on line AA were not available in the study report. This is significant because the only data available for wind speeds along the crest of the hill comes from two locations HT and CP. In addition, lacking data from the NE portion of line AA reduces the ability to compare the simulation results to the measured data on the leeward side of the hill.

4.3 Results of the Wind Resource Assessment

After establishing grid independence and validating the FLUENT simulation settings the resulting hub height velocity distribution can be analyzed to determine the feasibility of siting a wind farm in Poly Canyon. First, a simulation is run where the inlet power law velocity profile is based on an annual 24 hour average wind speed measured at the MET tower. The resulting hub height velocity profile is shown in Figure 41.

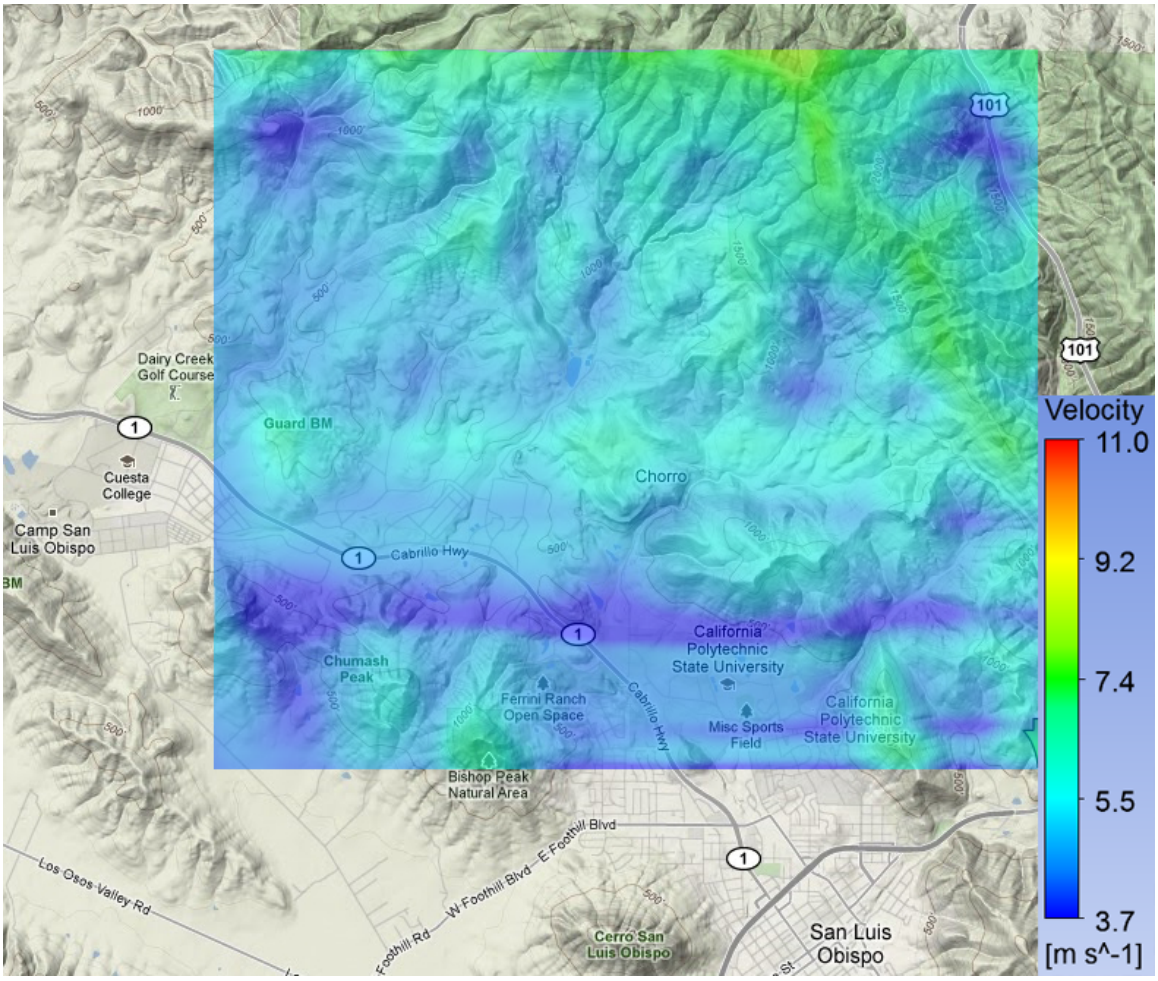


Figure 41: Hub height velocity distribution based on an annual 24 hour average wind speed.

It is difficult to tell from the velocity distribution in Figure 41 where the wind is at least Class III because much of the domain is near the minimum wind speed of 6.9m/s. To clarify which parts of the domain are at least Class III the hub height velocity distribution from Figure 41 is divided into wind class contours in Figure 42. Table 5 shows the velocity range for each IEC wind class at 80m. It is important to note that the IEC wind classes correspond to the wind resource and not wind turbines because the IEC also assigns classes to wind turbines however, those classes are designated by letters rather than numbers.

Table 5: Definition of IEC wind class velocity ranges at 80m.

IEC Wind Class	Velocity Range at 80m (m/s)
1	0 - 5.9
2	5.9 - 6.9
3	6.9 - 7.5
4	7.5 - 8.1
5	8.1 - 8.6
6	8.6 - 9.4
7	9.4 - 12.7

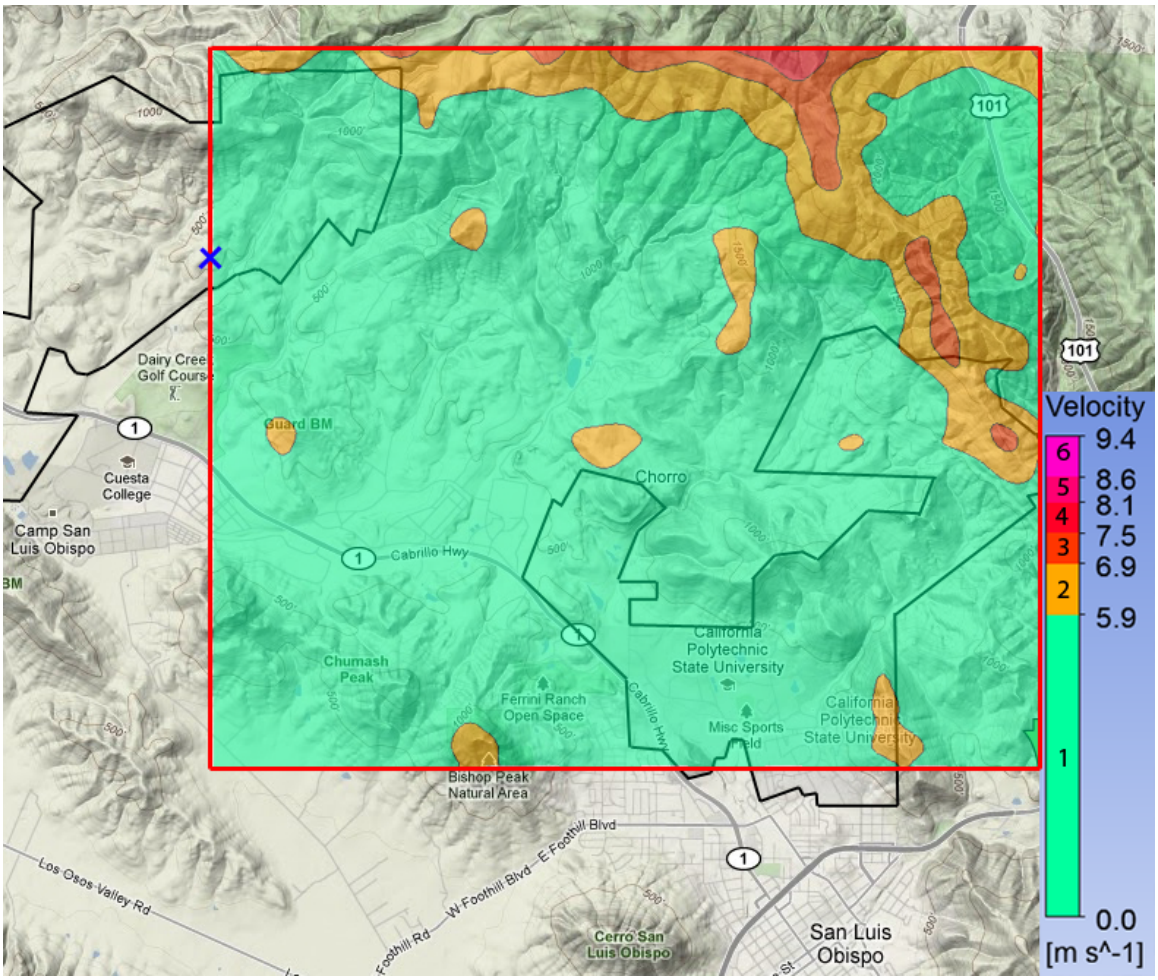


Figure 42: Velocity distribution from Figure 41 shown as contours based on wind class.

The boundaries of Cal Poly land are shown in black. The majority of Cal Poly land is a Class I resource which is not sufficient enough to warrant the construction of a wind farm however, there are small regions with Class II or III wind resources. However,

it is known from the data analysis above that the wind resource is much stronger during the daytime hours than at night. Based on this information another simulation is run where the inlet velocity profile is based on an annual average wind speed between the hours of 8am and 6pm. These hours were chosen because they represent the time of day when campus is the busiest and most energy is consumed. The hub height velocity profile corresponding to this simulation can be found in Figure 43.

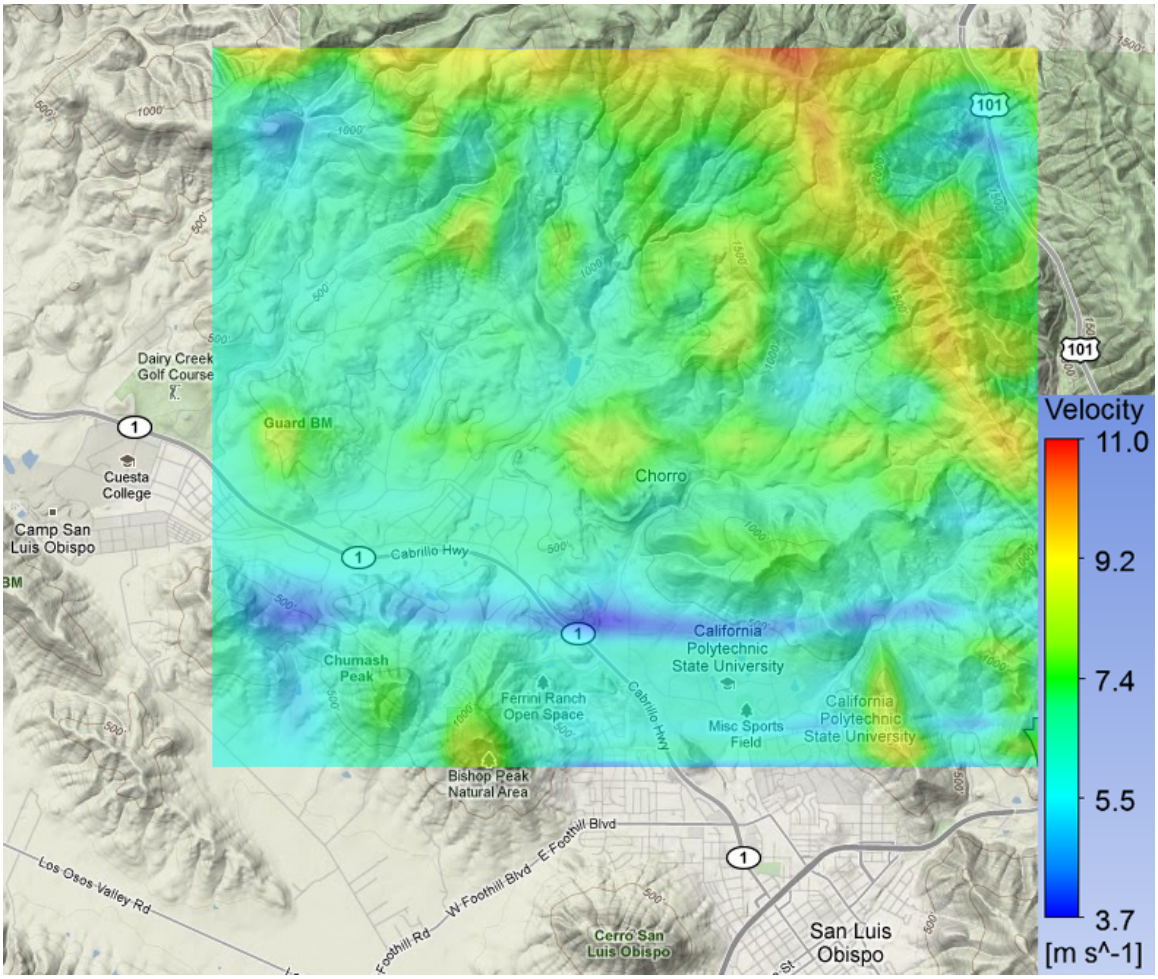


Figure 43: Hub height velocity distribution based on an annual daytime average wind speed.

Comparing Figures 41 and 43 it is clear that the wind resource during the daytime hours is much greater than the overall 24 hour resource. It is still difficult to discern if the

wind resource anywhere inside the area of interest is at least 6.9m/s so the velocity distribution from Figure 43 is shown as velocity contours in Figure 44.

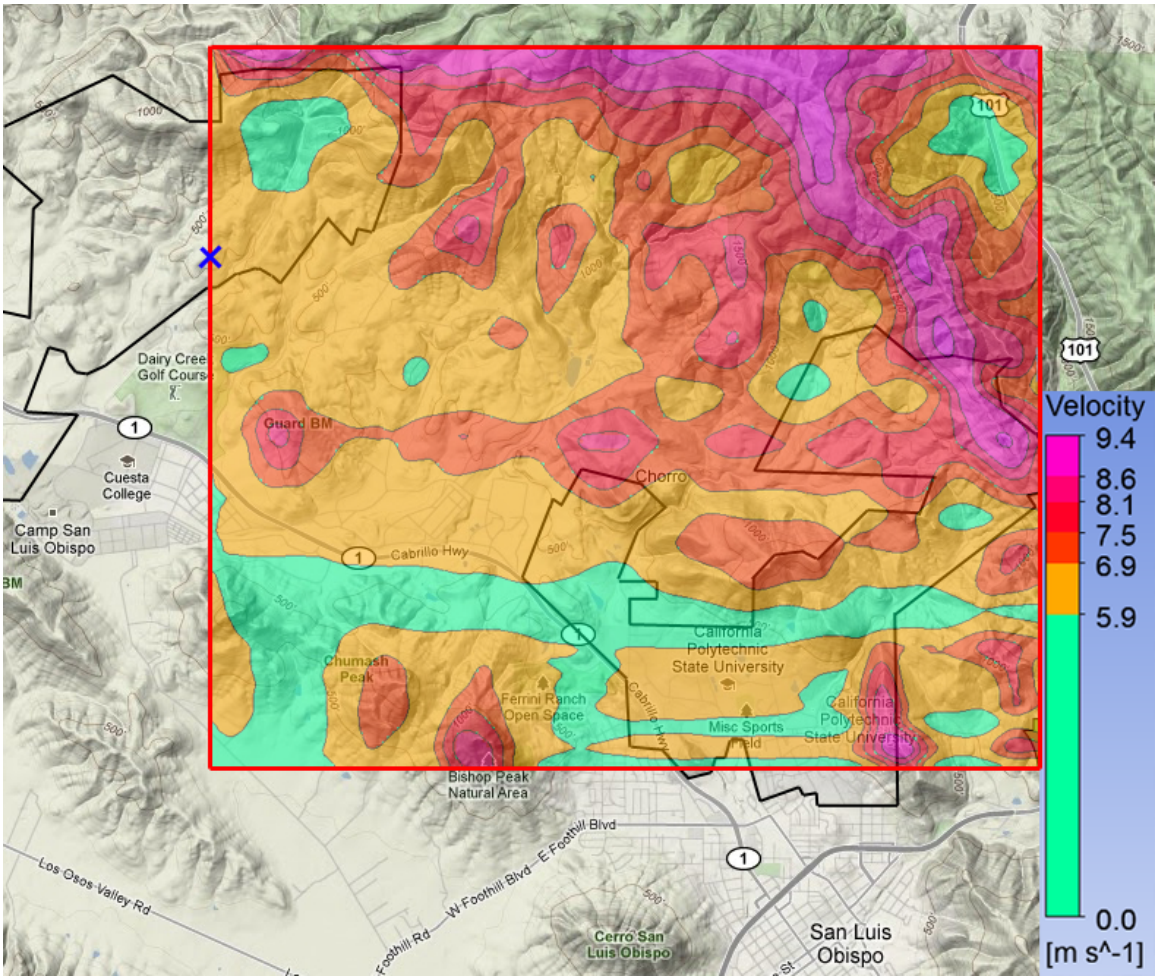


Figure 44: Velocity distribution from Figure 43 shown as velocity contours.

When only considering the daytime hours which are when campus is busiest and power consumption is greatest, it is clear that large portions of Cal Poly land have wind greater than 6.9m/s. Since only the daytime hours are being considered it is incorrect to say that the wind resource is of a certain class because IEC wind classes are defined based on annual 24 hour data. This is significant because although the daytime wind resource cannot be classified with an IEC wind class, large portions of Cal Poly land have wind which blows faster than 6.9m/s during the daytime on an annual basis.

Chapter 5: Conclusions and Recommendations

The results of the Cal Poly FLUENT CFD simulation suggest the wind resource in Poly Canyon has significant potential for power generation. On an annual 24 hour basis the majority of Cal Poly land is only an IEC Class I wind resource however there are small areas with a Class II or III wind resource. The MET tower at Escuela Ranch has a Class I wind resource. A Class I resource is not sufficient for power generation but the diurnal and seasonal wind patterns work in favor of the campus. The wind is strongest during the fall, winter, and spring when campus is populated and using the most energy. During the summer the wind is weaker but there is less energy consumed on campus during those months. In addition the wind blows harder during the daytime than at night. This is positive because much more energy is used on campus during the daytime than at night. On an annual basis when only considering the daytime hours from 8am to 6pm there are significant portions of Cal Poly land which have wind speeds above 6.9m/s. These areas have the potential to generate usable amounts of power at the same times when the campus is using the most power.

Before any further action is taken MET towers should be constructed in the areas with annual daytime wind speeds above 6.9m/s to physically verify the results of the CFD simulation. The areas of Cal Poly land with annual daytime wind speeds above 6.9m/s provide options for tower placement depending on whether it is desired for the towers to be visible or required for them to be hidden from campus and/or the city of San Luis Obispo. Figure 45 shows the recommended locations to install six MET towers for the purpose of validating this wind resource assessment. The recommended locations for the MET towers are depicted as black circles.

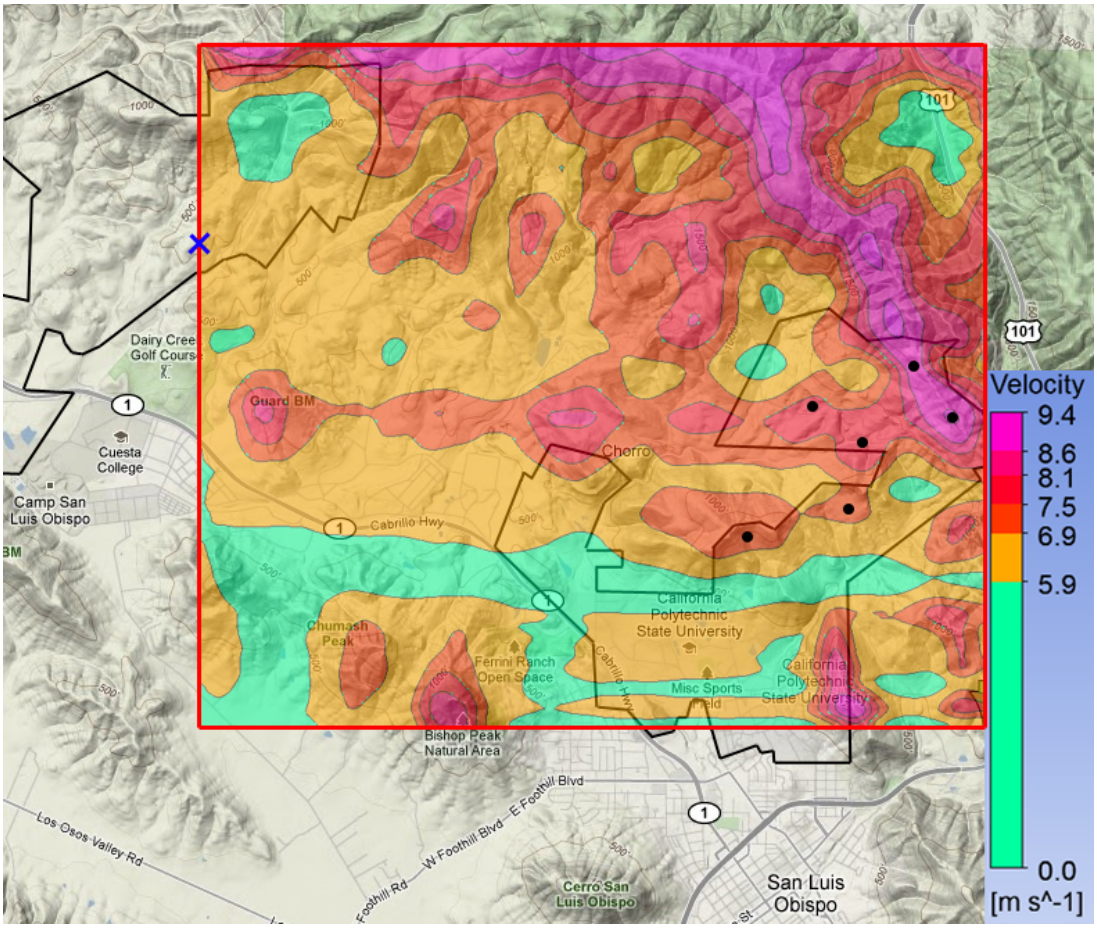


Figure 45: Recommended locations for installation of MET towers to validate the wind resource assessment.

Each newly constructed MET tower should measure wind speed in at least four different heights to provide an accurate representation of the velocity profile at that location. In addition, one of the measurement heights should be as close to hub height as possible and include a wind vane for direction measurements. These six MET tower locations were chosen because they are located in areas with varying average daytime wind speeds which will provide a more robust validation. In addition, the MET tower locations are unlikely to cause interference with each others wind speed and direction measurements since the wind blows predominately blows out of the west. Once the MET towers are built data should be collected for a minimum of 1 year before deciding whether to move forward with construction of a wind farm.

Appendix A: NRG Systems Anemometer Calibration Report

OTECH ENGINEERING INC.

ANEMOMETER CALIBRATION REPORT

Test Date: 8 November 2010 Revision No: 0

630 Peña Drive, Suite 800
Davis, CA 95618-7726
Office: (530) 757-2264
http://www.otechwind.com

Customer Information

NRG Systems, Inc.
110 Riggs Road
Hinesburg, VT 05461
USA

Instrument Under Test (IUT)

Model No: NRG #40 Sine
Serial No: 179500163914
Output: Sine Wave
IUT Power: 0 VDC
Heater Power: 0 VDC
Mount Diameter: 12.7 mm
Test Procedure: OTECH-CP-001

Wind Tunnel Test Facility

Otech Tunnel ID: WT1C
Type: Eiffel (open circuit, suction)
Test Section Size: 0.61 m x 0.61 m x 1.22 m
Manufacturer: Engineering Laboratory Design, Inc.

Data Acquisition

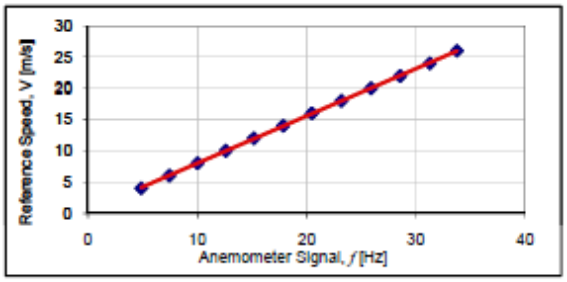
Hardware: National Instruments CDAQ-9172 USB 2.0 chassis with NI 9205 32-chan 16-bit AI module
Software: National Instruments LabVIEW 8.5
Signal Reduction Method for IUT: FFT Analysis

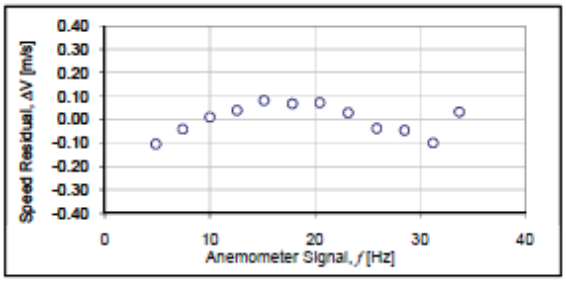
Measuring Equipment

Reference Speed: Four United Sensor Type PA Pitot-static tubes sensed by an MKS Barotron Type 220D Differential Pressure Transducer (NIST traceable)
Amb. Pressure: Setra Model 270 Barometer (NIST traceable)
Amb. Temperature: OMEGA HX94 SS Probe (NIST traceable)
Relative Humidity: OMEGA HX94 SS Probe (NIST traceable)

Test Conditions

Reference Speed Position Correction = 1
Reference Speed Blockage Correction = 1
Mean Ambient Pressure = 101,731 Pa
Mean Ambient Temperature = 23.0 deg C
Mean Relative Humidity = 50.2% RH
Mean Density = 1.1906 kg/cubic meter



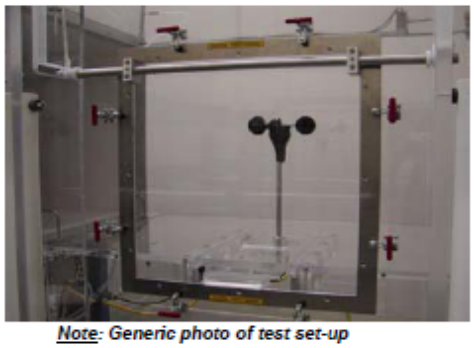


Transfer Function Test Results: **V [m/s] = 0.757 f [Hz] + 0.43**

Regression Parameters

r = 0.99996
std. err. estimate = 0.0676 m/s
slope = 0.757 m/s per Hz
std. err. slope = 0.00214 m/s per Hz
offset = 0.43 m/s
std. err. offset = 0.04556 m/s

Reference Speed [m/s]	Anemometer Output [Hz]	Residual [m/s]	Ref. Speed Uncertainty
3.996	4.856	-0.104	0.544%
8.001	9.999	0.010	0.522%
11.979	15.161	0.081	0.477%
15.982	20.463	0.072	0.470%
19.974	25.883	-0.037	0.483%
23.975	31.254	-0.100	0.485%
25.993	33.746	0.032	0.468%
21.976	28.540	-0.046	0.469%
17.986	23.169	0.029	0.481%
13.995	17.844	0.067	0.482%
9.993	12.593	0.039	0.479%
6.001	7.422	-0.041	0.509%



***Note**: Generic photo of test set-up*

Approved by: Adam Havner, Lab Manager Adam Havner

This document reports that the above IUT was tested at Otech Engineering, Inc., a wind tunnel laboratory accredited in accordance with the recognised International Standard ISO/IEC 17025:2005 (Certificate number CL-126). This accreditation demonstrates technical competence for a defined scope and the operation of a laboratory quality management system (refer joint ISO-ILAC-IAF Communique dated January 2009). Uncertainties estimated at 95 % confidence level. This report shall not be reproduced except in full, without written approval from Otech Engineering, Inc.

IAS ACCREDITED

References available upon request. Page 1 of 1 179500163914_2010-11-08.pdf

Appendix B: Second Wind Anemometer Data Sheet

Model C3 Anemometer Features

A new choice with uncompromising accuracy:

Second Wind is pleased to provide you with an improved version of the popular three-cup design used in wind assessments for decades. Both our calibrated and uncalibrated versions are manufactured to precise industry standards.

Calibrated

Our Model C3C anemometer has been calibrated at the highly respected Massachusetts Institute of Technology's Wright Brothers wind tunnel.

Uncalibrated

Customers with their own calibration capability can now choose this option.

For more information:

Visit www.secondwind.com to learn about this new product line:

#981 Second Wind C3 Anemometer w/ Boot

#982 Second Wind Calibrated C3C Anemometer w/ Boot

Tougher materials stand up to the toughest environments

The Model C3 rotor is made of tough polycarbonate for exceptional durability and reliability. The Model C3 sensor base is also made of rugged polycarbonate, making it more resistant to damage during installation.

- Manufactured by Second Wind with exceptional quality control—all units are tested mechanically and electronically before shipping.
- A distinctive blue vinyl boot shields wiring for long-term performance.
- The C3 is manufactured to meet new standards and is RoHS-compliant—no toxic metals.

Easier tracking

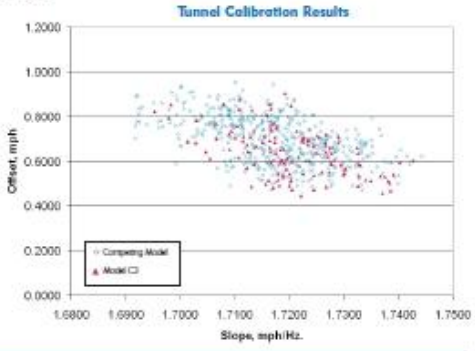
All models are individually laser engraved with serial number and date code, simplifying tracking and data analysis.

Specifications:

- Conical cups measure 51 mm (2 inches) in diameter
- Rotor diameter is 190 mm (7.5 inches)
- Standard AC output, frequency proportional to cup rotational speed
- Shielded AC pickup coil, 4100 turns of #41 wire
- Four-pole Indox 1 magnet rotates with the cup assembly
- Fully hardened beryllium-copper shaft running in self-lubricating modified Teflon bearings, with protective boot to make the system dirt and water resistant
- Rated bearing PV (pressure-velocity) factor is 20,000
  - At 15 mph PV is approx. 500.
  - At 100 mph PV is approx. 2,000.
- Rotor assembly moment of inertia = 68 x 10^{-6} S-ft² (or 92.2 x 10^{-6} kg-m²)
- Distance constant = 10 feet (3.0 meters)
- Transfer Function: m/s = (Hz x 0.766) + 0.324
  [miles per hour = (Hz x 1.714) + 0.725]
- Accuracy: within 0.1 m/s (0.2 mph) for the range 5 m/s to 25 m/s (11 mph to 55 mph)

Performance validated

Multiple tests—including performance data from hundreds of C3 units already in the field—and detailed wind tunnel comparisons involving hundreds of units—prove that the C3's performance is virtually identical to the industry's top-selling product.



SECONDWIND
Measurement. Management. And more.

Second Wind Inc. 366 Summer Street, Somerville, MA 02144 USA
Tel. 617 776-8520 • Fax 617 776-0391 • www.secondwind.com

Appendix C: Table of Calibration Curve Fit Coefficients

Generic curve fit equation for *y* as a function of *x*:

$$y = C_6x^6 + C_5x^5 + C_4x^4 + C_3x^3 + C_2x^2 + C_1x + C_0$$

Accompanying Figure	C_0	C_1	C_2	C_3	C_4	C_5	C_6
6	-	1.437	4.769E-02	-2.268E-03	5.236E-05	-4.694E-07	-
8	-2.428	3.104	-1.941E-01	1.267E-02	-4.249E-04	7.106E-06	-4.716E-08
11	1.777	1.368	3.924E-02	-1.918E-03	4.170E-05	-3.388E-07	-
12	2.362	1.068	8.051E-02	-3.997E-03	8.702E-05	-7.027E-07	-
13	1.682	1.453	3.039E-03	2.447E-03	-1.588E-04	3.618E-06	-2.842E-08
14	7.884	-7.045	2.358	-2.407E-01	1.189E-02	-2.791E-04	2.474E-06
15	1.079	1.686	-	-	-	-	-

Appendix D: Power Law User Defined Function

```
#include "udf.h"

#define Ur 3.59
#define Yr 24.384

DEFINE_PROFILE(x_velocity,t,i)
{
real z[ND_ND];
real y;
face_t f;

begin_f_loop(f,t)
{
F_CENTROID(z,f,t);
y = z[1];
F_PROFILE(f,t,i) = Ur*pow((y/Yr),(1./7.));
}
end_f_loop(f,t)
}
```

Appendix E: MATLAB Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  MET Tower Data Analysis Script               %
%  Written by: Jason Smith                      %
%  Written for: Cal Poly ME Dept.               %
%  Thesis Project: Winter Quarter 2011          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%Clears the workspace of any variables left over from previous use
%for a fresh start
clear all;
clc;

%Reads the date from the system clock and saves it as a vector of the
%form [year month day hour min sec]
datevector = fix(clock);

%Checks if the current date resides within a leap year. ly = 1 if yes
or 0
%if no
ly = leapyear(datevector(1));

%Removing the time from the date vector. The date vector now is of the
form
%[year month day]
datevector = [datevector(1) datevector(2) datevector(3)];

%Modifies the date for the purpose of downloading the correct wind
data.
%One day is subtracted from the date vector so that the most recent
data
%can be downloaded. The following if loop checks if todays date is the
1st
%of the month and if so changes the day to the proper number depending
on
%what month it is. If today is not the first of the month 1 day is
simply
%subtracted from todays date.
if datevector(3) == 1
    switch (datevector(2))
        case {1},
            datevector(1) = (datevector(1)-1);
            datevector(2) = 12;
            datevector(3) = 31;
        case {2,4,6,8,9,11},
            datevector(2) = (datevector(2)-1);
            datevector(3) = 31;
        case {3},
            datevector(2) = (datevector(2)-1);
            datevector(3) = 28 + ly;
        case {5,7,10,12},
            datevector(2) = (datevector(2)-1);
            datevector(3) = 30;
```

```
    end
else
    datevector(3) = (datevector(3)-1);
end

%Creates strings specifying which data to download and where to store
it.
%The switch loops account for whether the month and day are 1-9 or 10-
31.
%This is necessary because on the website where the data is being
%downloaded, numbers 1-9 have a leading 0 in front of them.
switch (datevector(2))
    case {1,2,3,4,5,6,7,8,9},
        downfile = ['http://me.me.calpoly.edu:1280/wind/archives/wind-'
int2str(datevector(1)) '-0' int2str(datevector(2))];
        windfile = ['T:/Cal_Poly_Wind_Data/wind-'
int2str(datevector(1)) '-0' int2str(datevector(2))];
    case {10,11,12},
        downfile = ['http://me.me.calpoly.edu:1280/wind/archives/wind-'
int2str(datevector(1)) '-' int2str(datevector(2))];
        windfile = ['T:/Cal_Poly_Wind_Data/wind-'
int2str(datevector(1)) '-' int2str(datevector(2))];
end

switch (datevector(3))
    case {1,2,3,4,5,6,7,8,9},
        fulldownfile = [downfile '-0' int2str(datevector(3)) '.zip'];
        fullwindfile = [windfile '-0' int2str(datevector(3)) '.xlsx'];
    case
{10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31},
        fulldownfile = [downfile '-' int2str(datevector(3)) '.zip'];
        fullwindfile = [windfile '-' int2str(datevector(3)) '.xlsx'];
end

%Downloads and unzips the data, then stores it to the location in
purple
%type
unzip(fulldownfile,'T:/Cal_Poly_Wind_Data/');

%Converting the date from MATLAB datum to EXCEL datum. This is
necessary
%because all of the calculations and filtering (i.e. calculating wind
speed
%from rotations and directions from sines and cosines) will be
performed in
%Excel.
mdate = datenum(datevector);
edate = mdate - datenum('30-dec-1899');

%Reads the data from the downloaded and unzipped .txt file and creates
a
%cell array called "rawdata".
fid = fopen('T:/Cal_Poly_Wind_Data/wind-today.txt');
rawdata = textscan(fid, '%s %s %d %d %d %d %d %d');
fclose(fid);
```

```
%Cell arrays are difficult to work with because they can hold a
combination
%of strings, integers, etc. so this is creating a normal array from
each
%column of the cell array called "rawdata". The xlswrite and xlsread on
the
%first two lines is an easy way of converting the time string
%(i.e. 12:34:56) to a value between zero and 1 which is how Excel
%interprets times.
xlswrite(fullwindfile, rawdata{2}, 'Sheet2', 'A1');
datatime = xlsread(fullwindfile, 'Sheet2');
data3 = double(rawdata{3});
data4 = double(rawdata{4});
data5 = double(rawdata{5});
data6 = double(rawdata{6});
data7 = double(rawdata{7});
data8 = double(rawdata{8});

%Making sure all columns are same length and if not, removing the data
from
%the incomplete line. This is important because many times the final
line
%of the downloaded .txt file is incomplete which was causing numerous
%errors throughout the rest of the code.
size2 = numel(datatime);
size3 = numel(data3);
size4 = numel(data4);
size5 = numel(data5);
size6 = numel(data6);
size7 = numel(data7);
size8 = numel(data8);
sizevector = [size2 size3 size4 size5 size6 size7 size8];
minsize = min(sizevector);
datatime = datatime(1:minsize,1);
data3 = data3(1:minsize,1);
data4 = data4(1:minsize,1);
data5 = data5(1:minsize,1);
data6 = data6(1:minsize,1);
data7 = data7(1:minsize,1);
data8 = data8(1:minsize,1);

%Creating a column vector containing the date pertaining to data. This
is
%important because the downloaded .txt file does not contain any date
data
%so this will be needed for filtering by day later in the code.
for z = 1:minsize
    datadate(z,1) = edate;
end

%A matrix called "datamatrix" is created by appending all of the column
%vectors created above together. Then the data is sorted by tower
height
%and saved as a matrix "sorteddata" as well as written to its own Excel
%file (i.e. wind-2011-01-18) in case it is needed in the future.
```

```
datamatrix =
cat(2,data3,datadate,datatime,data4,data5,data6,data7,data8);
sorteddata = sortrows(datamatrix);
xlswrite(fullwindfile, sorteddata, 'Sheet1', 'A1');

%Each while loop is creating a counter which corresponds to the rows in
the
%matrix "sorteddata" for each different tower height.
counter00 = 1;
while (sorteddata(counter00,1) <= 11)
    counter00 = counter00 + 1;
end
counter00 = counter00 - 1;

counter12 = 1;
while (sorteddata(counter12,1) <= 12)
    counter12 = counter12 + 1;
end
counter12 = counter12 - 1;

counter14 = 1;
while (sorteddata(counter14,1) <= 14)
    counter14 = counter14 + 1;
end
counter14 = counter14 - 1;

counter16 = 1;
while (sorteddata(counter16,1) <= 16)
    counter16 = counter16 + 1;
end
counter16 = counter16 - 1;

counter18 = 1;
while (minsize >= counter18) && (sorteddata(counter18,1) <= 18)
    counter18 = counter18 + 1;
end
counter18 = counter18 - 1;

%The counters are then used to chop up the matrix "sorteddata" into 4
%different matrices each containing data from one of the tower heights
%(i.e. data12 is a matrix corresponding to only the data from the 20ft
%tower height.)
data12 = sorteddata(counter00 + 1:counter12,1:6);
data14 = sorteddata(counter12 + 1:counter14,1:6);
data16 = sorteddata(counter14 + 1:counter16,:);
data18 = sorteddata(counter16 + 1:counter18,1:6);

% %Converting anemometer revolutions per ten second interval to wind
speed
% %using transfer functions found during wind tunnel calibrations.
% data12(:,4:6) = data12(:,4:6)./10;
% data12(:,4:6) = 1.685.*data12(:,4:6)+1.212;
%
% data14(:,4:6) = data14(:,4:6)./10;
% data14(:,4:6) = 1.691.*data14(:,4:6)+1.409;
```

```
%
% data16(:,4:6) = data16(:,4:6)./10;
% data16(:,4:6) = 1.696.*data16(:,4:6)+1.302;
%
% data18(:,4:6) = data18(:,4:6)./10;
% data18(:,4:6) = 1.692.*data18(:,4:6)+1.215;

%Converting anemometer revolutions per ten second interval to wind
speed
%using transfer functions found during wind tunnel calibrations.
data12(:,4:6) = data12(:,4:6)./10;
data12(:,4:6) = -3.388E-7.*data12(:,4:6).^5+4.16967E-
5.*data12(:,4:6).^4-1.91809E-3.*data12(:,4:6).^3+3.92376E-
2.*data12(:,4:6).^2+1.36816.*data12(:,4:6)+1.77746;

data14(:,4:6) = data14(:,4:6)./10;
data14(:,4:6) = -7.02713E-7.*data14(:,4:6).^5+8.70192E-
5.*data14(:,4:6).^4-3.99707E-3.*data14(:,4:6).^3+8.05102E-
2.*data14(:,4:6).^2+1.06780.*data14(:,4:6)+2.36219;

data16(:,4:6) = data16(:,4:6)./10;
data16(:,4:6) = -2.84187E-8.*data16(:,4:6).^6+3.61786E-
6.*data16(:,4:6).^5-1.58839E-4.*data16(:,4:6).^4+2.44651E-
3.*data16(:,4:6).^3+3.03908E-
2.*data16(:,4:6).^2+1.45282.*data16(:,4:6)+1.68249;

data18(:,4:6) = data18(:,4:6)./10;
data18(:,4:6) = -4.34505E-7.*data18(:,4:6).^5+5.10057E-
5.*data18(:,4:6).^4-2.29264E-3.*data18(:,4:6).^3+4.77199E-
2.*data18(:,4:6).^2+1.27802.*data18(:,4:6)+2.06891;

%Removing data where the average or max wind speed is obviously
incorrect
%(i.e. >100mph). Im calling a custom function I wrote called
"removedata2".
limit = 100;
[data12] = removedata2(data12,limit);
[data14] = removedata2(data14,limit);
[data16] = removedata2(data16,limit);
[data18] = removedata2(data18,limit);

%Calculating wind direction for 60ft tower height and checks if the
value
%is greater than 360 degrees. If so 360 degrees is subtracted so that
all
%direction values are between 0 and 360 degrees. This creates a vector
%called "direction".
sines = double(data16(:,8));
cosines = double(data16(:,7));
direction = (atan2(sines,cosines)).*180/pi()+236;
direction = uint16(direction);
if (direction > 360)
    direction = direction - 360;
end
```

```
%The vector "append" is being read from an Excel file called
%append-counters. This vector contains values corresponding to the
number
%of rows for each tower height in the master Excel file which contains
all
%the wind data. That way the newly downloaded and processed data can be
%appended to the bottom of the Excel file containing all the wind data.
append = xlsread('T:/Cal_Poly_Wind_Data/append-counters.xlsx');

%Creating number variables based on the values in the "append" vector.
num18 = append(1,1)+1;
num16 = append(2,1)+1;
num14 = append(3,1)+1;
num12 = append(4,1)+1;

%Creating strings from the number variables. These strings will tell
Excel
%which cells to write the newly processed data to.
cell18 = ['A' int2str(num18)];
cell16 = ['A' int2str(num16)];
celld = ['F' int2str(num16)];
cell14 = ['A' int2str(num14)];
cell12 = ['A' int2str(num12)];

%These few lines are the ones which actually append the newly processed
%data to the master data file in Excel.
xlswrite('T:/Cal_Poly_Wind_Data/wind-data.xlsx', data18(:,2:6),
'Tower18', cell18);
xlswrite('T:/Cal_Poly_Wind_Data/wind-data.xlsx', data16(:,2:6),
'Tower16', cell16);
xlswrite('T:/Cal_Poly_Wind_Data/wind-data.xlsx', direction, 'Tower16',
celld);
xlswrite('T:/Cal_Poly_Wind_Data/wind-data.xlsx', data14(:,2:6),
'Tower14', cell14);
xlswrite('T:/Cal_Poly_Wind_Data/wind-data.xlsx', data12(:,2:6),
'Tower12', cell12);

%After the new data has been written to Excel these four lines read all
of
%the data from the master Excel file. One line is used for each
different
%tower height.
tower18 = xlsread('T:/Cal_Poly_Wind_Data/wind-data.xlsx', 'Tower18');
tower16 = xlsread('T:/Cal_Poly_Wind_Data/wind-data.xlsx', 'Tower16');
tower14 = xlsread('T:/Cal_Poly_Wind_Data/wind-data.xlsx', 'Tower14');
tower12 = xlsread('T:/Cal_Poly_Wind_Data/wind-data.xlsx', 'Tower12');

%These four lines calculate the number of rows in the master Excel file
for
%each different tower height. This is for the purpose of updating the
%"append-counters" Excel file so next time the script is run the data
will
%be appended into the proper rows of the master Excel data file.
t18num = numel(tower18(:,1));
t16num = numel(tower16(:,1));
t14num = numel(tower14(:,1));
```

```
t12num = numel(tower12(:,1));

%This is actually creating the "append" vector from the four lines
above
%and writing it to the Excel file "append-counters".
append = [t18num; t16num; t14num; t12num];
xlswrite('T:/Cal_Poly_Wind_Data/append-counters.xlsx', append);

%Now that MATLAB has read all of the data in from the master Excel data
%file, matrices are created for each different tower height.
%(i.e. tower18 corresponds to all of the data we have dating all the
way
%back to 4-1-2009 for the 80ft tower height.) Then each matrix is
sorted by
%date in descending order so that the newest data is at the top and
then
%sorted within each date by time in ascending order.
tower18 = sortrows(tower18,[-1,2]);
tower16 = sortrows(tower16,[-1,2]);
tower14 = sortrows(tower14,[-1,2]);
tower12 = sortrows(tower12,[-1,2]);

%Each while loop is creating a counter which corresponds to the rows in
%each matrix "towerXX" for a certain period of time.
%1 day counters
counter18d = 1;
while (t18num >= counter18d) && (tower18(counter18d,1) == edate)
    counter18d = counter18d + 1;
end
counter18d = counter18d - 1;

counter16d = 1;
while (t16num >= counter16d) && (tower16(counter16d,1) == edate)
    counter16d = counter16d + 1;
end
counter16d = counter16d - 1;

counter14d = 1;
while (t14num >= counter14d) && (tower14(counter14d,1) == edate)
    counter14d = counter14d + 1;
end
counter14d = counter14d - 1;

counter12d = 1;
while (t12num >= counter12d) && (tower12(counter12d,1) == edate)
    counter12d = counter12d + 1;
end
counter12d = counter12d - 1;

%7 day counters
counter18w = 1;
while (t18num >= counter18w) && (tower18(counter18w,1) >= (edate-6))
    counter18w = counter18w + 1;
end
counter18w = counter18w - 1;
```

```
counter16w = 1;
while (t16num >= counter16w) && (tower16(counter16w,1) >= (edate-6))
    counter16w = counter16w + 1;
end
counter16w = counter16w - 1;

counter14w = 1;
while (t14num >= counter14w) && (tower14(counter14w,1) >= (edate-6))
    counter14w = counter14w + 1;
end
counter14w = counter14w - 1;

counter12w = 1;
while (t12num >= counter12w) && (tower12(counter12w,1) >= (edate-6))
    counter12w = counter12w + 1;
end
counter12w = counter12w - 1;

%30 day counters
counter18m = 1;
while (t18num >= counter18m) && (tower18(counter18m,1) >= (edate-29))
    counter18m = counter18m + 1;
end
counter18m = counter18m - 1;

counter16m = 1;
while (t16num >= counter16m) && (tower16(counter16m,1) >= (edate-29))
    counter16m = counter16m + 1;
end
counter16m = counter16m - 1;

counter14m = 1;
while (t14num >= counter14m) && (tower14(counter14m,1) >= (edate-29))
    counter14m = counter14m + 1;
end
counter14m = counter14m - 1;

counter12m = 1;
while (t12num >= counter12m) && (tower12(counter12m,1) >= (edate-29))
    counter12m = counter12m + 1;
end
counter12m = counter12m - 1;

%365 day counters
counter18y = 1;
while (t18num >= counter18y) && (tower18(counter18y,1) >= (edate-364))
    counter18y = counter18y + 1;
end
counter18y = counter18y - 1;

counter16y = 1;
while (t16num >= counter16y) && (tower16(counter16y,1) >= (edate-364))
    counter16y = counter16y + 1;
end
```

```
counter16y = counter16y - 1;

counter14y = 1;
while (t14num >= counter14y) && (tower14(counter14y,1) >= (edate-364))
    counter14y = counter14y + 1;
end
counter14y = counter14y - 1;

counter12y = 1;
while (t12num >= counter12y) && (tower12(counter12y,1) >= (edate-364))
    counter12y = counter12y + 1;
end
counter12y = counter12y - 1;

%Creating new matrices by parsing data into different durations.
%Notation: the two digits after the word tower correspond to the tower
%height and the letter after the two digits is as follows.
%(d= 1 day, w= 7 days, m= 30 days, y= 365 days)
tower18d = tower18(1:counter18d,:);
tower16d = tower16(1:counter16d,:);
tower14d = tower14(1:counter14d,:);
tower12d = tower12(1:counter12d,:);

tower18w = tower18(1:counter18w,:);
tower16w = tower16(1:counter16w,:);
tower14w = tower14(1:counter14w,:);
tower12w = tower12(1:counter12w,:);

tower18m = tower18(1:counter18m,:);
tower16m = tower16(1:counter16m,:);
tower14m = tower14(1:counter14m,:);
tower12m = tower12(1:counter12m,:);

tower18y = tower18(1:counter18y,:);
tower16y = tower16(1:counter16y,:);
tower14y = tower14(1:counter14y,:);
tower12y = tower12(1:counter12y,:);

%Then once the data has been divided up into the different desired
%durations, it must be filtered by hour. These lines are calling a
custom
%function I wrote called "hourfilter" which filters the date-wise
parsed
%data by hour.
%Notation: (t stands for tower, next two digits correspond to tower
height,
%next character is either d,w,m, or y for the date-wise parsing, and
the
%last two digits correspond to the time. 00=midnight to 1am, 01=1am to
2am,
%and so on.) Each of these is a matrix corresponding to one hour of the
%date-wise parsed data.
%1 day data
```

```
[t18d00,t18d01,t18d02,t18d03,t18d04,t18d05,t18d06,t18d07,t18d08,t18d09,
t18d10,t18d11,t18d12,t18d13,t18d14,t18d15,t18d16,t18d17,t18d18,t18d19,t
18d20,t18d21,t18d22,t18d23] = hourfilter (tower18d(:,2:5));
[t16d00,t16d01,t16d02,t16d03,t16d04,t16d05,t16d06,t16d07,t16d08,t16d09,
t16d10,t16d11,t16d12,t16d13,t16d14,t16d15,t16d16,t16d17,t16d18,t16d19,t
16d20,t16d21,t16d22,t16d23] = hourfilter (tower16d(:,2:6));
[t14d00,t14d01,t14d02,t14d03,t14d04,t14d05,t14d06,t14d07,t14d08,t14d09,
t14d10,t14d11,t14d12,t14d13,t14d14,t14d15,t14d16,t14d17,t14d18,t14d19,t
14d20,t14d21,t14d22,t14d23] = hourfilter (tower14d(:,2:5));
[t12d00,t12d01,t12d02,t12d03,t12d04,t12d05,t12d06,t12d07,t12d08,t12d09,
t12d10,t12d11,t12d12,t12d13,t12d14,t12d15,t12d16,t12d17,t12d18,t12d19,t
12d20,t12d21,t12d22,t12d23] = hourfilter (tower12d(:,2:5));

%1 week data
[t18w00,t18w01,t18w02,t18w03,t18w04,t18w05,t18w06,t18w07,t18w08,t18w09,
t18w10,t18w11,t18w12,t18w13,t18w14,t18w15,t18w16,t18w17,t18w18,t18w19,t
18w20,t18w21,t18w22,t18w23] = hourfilter (tower18w(:,2:5));
[t16w00,t16w01,t16w02,t16w03,t16w04,t16w05,t16w06,t16w07,t16w08,t16w09,
t16w10,t16w11,t16w12,t16w13,t16w14,t16w15,t16w16,t16w17,t16w18,t16w19,t
16w20,t16w21,t16w22,t16w23] = hourfilter (tower16w(:,2:6));
[t14w00,t14w01,t14w02,t14w03,t14w04,t14w05,t14w06,t14w07,t14w08,t14w09,
t14w10,t14w11,t14w12,t14w13,t14w14,t14w15,t14w16,t14w17,t14w18,t14w19,t
14w20,t14w21,t14w22,t14w23] = hourfilter (tower14w(:,2:5));
[t12w00,t12w01,t12w02,t12w03,t12w04,t12w05,t12w06,t12w07,t12w08,t12w09,
t12w10,t12w11,t12w12,t12w13,t12w14,t12w15,t12w16,t12w17,t12w18,t12w19,t
12w20,t12w21,t12w22,t12w23] = hourfilter (tower12w(:,2:5));

%30 day data
[t18m00,t18m01,t18m02,t18m03,t18m04,t18m05,t18m06,t18m07,t18m08,t18m09,
t18m10,t18m11,t18m12,t18m13,t18m14,t18m15,t18m16,t18m17,t18m18,t18m19,t
18m20,t18m21,t18m22,t18m23] = hourfilter (tower18m(:,2:5));
[t16m00,t16m01,t16m02,t16m03,t16m04,t16m05,t16m06,t16m07,t16m08,t16m09,
t16m10,t16m11,t16m12,t16m13,t16m14,t16m15,t16m16,t16m17,t16m18,t16m19,t
16m20,t16m21,t16m22,t16m23] = hourfilter (tower16m(:,2:6));
[t14m00,t14m01,t14m02,t14m03,t14m04,t14m05,t14m06,t14m07,t14m08,t14m09,
t14m10,t14m11,t14m12,t14m13,t14m14,t14m15,t14m16,t14m17,t14m18,t14m19,t
14m20,t14m21,t14m22,t14m23] = hourfilter (tower14m(:,2:5));
[t12m00,t12m01,t12m02,t12m03,t12m04,t12m05,t12m06,t12m07,t12m08,t12m09,
t12m10,t12m11,t12m12,t12m13,t12m14,t12m15,t12m16,t12m17,t12m18,t12m19,t
12m20,t12m21,t12m22,t12m23] = hourfilter (tower12m(:,2:5));

%1 year data
[t18y00,t18y01,t18y02,t18y03,t18y04,t18y05,t18y06,t18y07,t18y08,t18y09,
t18y10,t18y11,t18y12,t18y13,t18y14,t18y15,t18y16,t18y17,t18y18,t18y19,t
18y20,t18y21,t18y22,t18y23] = hourfilter (tower18y(:,2:5));
[t16y00,t16y01,t16y02,t16y03,t16y04,t16y05,t16y06,t16y07,t16y08,t16y09,
t16y10,t16y11,t16y12,t16y13,t16y14,t16y15,t16y16,t16y17,t16y18,t16y19,t
16y20,t16y21,t16y22,t16y23] = hourfilter (tower16y(:,2:6));
[t14y00,t14y01,t14y02,t14y03,t14y04,t14y05,t14y06,t14y07,t14y08,t14y09,
t14y10,t14y11,t14y12,t14y13,t14y14,t14y15,t14y16,t14y17,t14y18,t14y19,t
14y20,t14y21,t14y22,t14y23] = hourfilter (tower14y(:,2:5));
[t12y00,t12y01,t12y02,t12y03,t12y04,t12y05,t12y06,t12y07,t12y08,t12y09,
t12y10,t12y11,t12y12,t12y13,t12y14,t12y15,t12y16,t12y17,t12y18,t12y19,t
12y20,t12y21,t12y22,t12y23] = hourfilter (tower12y(:,2:5));

%Calculating mean wind speed, std. dev., and turbulence intensity for
```

```
%date and time-wise parsed data. This is done using another custom
function
%I wrote called "meanstdti". Each of the outputs from the function
which
%are the five terms in brackets on the far left of each line correspnds
to
%a [1X24] column vector where each value corresponds to one hour.
%Notation: (The t stands for tower, the next two digits correspond to
tower
%height, the next character is the date-wise duration, and the last
%character is m= mean, s= std. dev., and t= turbulence intensity.)
%The two of the outputs for each line have a trailing "n" character
these
%correspond to the same vectors without the "n" but have removed all of
the
%rows which are "NaN". For vector calculations MATLAB inserts "Nan"
when
%data is missing or divide by 0 etc. Removing these values from the
vectors
%becomes important later when plotting the data.
%1 day data
[t18dm,t18ds,t18dt,t18dmn,t18dsn] = meanstdti
(t18d00,t18d01,t18d02,t18d03,t18d04,t18d05,t18d06,t18d07,t18d08,t18d09,
t18d10,t18d11,t18d12,t18d13,t18d14,t18d15,t18d16,t18d17,t18d18,t18d19,t
18d20,t18d21,t18d22,t18d23);
[t16dm,t16ds,t16dt,t16dmn,t16dsn] = meanstdti
(t16d00,t16d01,t16d02,t16d03,t16d04,t16d05,t16d06,t16d07,t16d08,t16d09,
t16d10,t16d11,t16d12,t16d13,t16d14,t16d15,t16d16,t16d17,t16d18,t16d19,t
16d20,t16d21,t16d22,t16d23);
[t14dm,t14ds,t14dt,t14dmn,t14dsn] = meanstdti
(t14d00,t14d01,t14d02,t14d03,t14d04,t14d05,t14d06,t14d07,t14d08,t14d09,
t14d10,t14d11,t14d12,t14d13,t14d14,t14d15,t14d16,t14d17,t14d18,t14d19,t
14d20,t14d21,t14d22,t14d23);
[t12dm,t12ds,t12dt,t12dmn,t12dsn] = meanstdti
(t12d00,t12d01,t12d02,t12d03,t12d04,t12d05,t12d06,t12d07,t12d08,t12d09,
t12d10,t12d11,t12d12,t12d13,t12d14,t12d15,t12d16,t12d17,t12d18,t12d19,t
12d20,t12d21,t12d22,t12d23);

%1 week data
[t18wm,t18ws,t18wt,t18wmn,t18wsn] = meanstdti
(t18w00,t18w01,t18w02,t18w03,t18w04,t18w05,t18w06,t18w07,t18w08,t18w09,
t18w10,t18w11,t18w12,t18w13,t18w14,t18w15,t18w16,t18w17,t18w18,t18w19,t
18w20,t18w21,t18w22,t18w23);
[t16wm,t16ws,t16wt,t16wmn,t16wsn] = meanstdti
(t16w00,t16w01,t16w02,t16w03,t16w04,t16w05,t16w06,t16w07,t16w08,t16w09,
t16w10,t16w11,t16w12,t16w13,t16w14,t16w15,t16w16,t16w17,t16w18,t16w19,t
16w20,t16w21,t16w22,t16w23);
[t14wm,t14ws,t14wt,t14wmn,t14wsn] = meanstdti
(t14w00,t14w01,t14w02,t14w03,t14w04,t14w05,t14w06,t14w07,t14w08,t14w09,
t14w10,t14w11,t14w12,t14w13,t14w14,t14w15,t14w16,t14w17,t14w18,t14w19,t
14w20,t14w21,t14w22,t14w23);
[t12wm,t12ws,t12wt,t12wmn,t12wsn] = meanstdti
(t12w00,t12w01,t12w02,t12w03,t12w04,t12w05,t12w06,t12w07,t12w08,t12w09,
t12w10,t12w11,t12w12,t12w13,t12w14,t12w15,t12w16,t12w17,t12w18,t12w19,t
12w20,t12w21,t12w22,t12w23);
```

```
%30 day data
[t18mm,t18ms,t18mt,t18mmn,t18msn] = meanstdti
(t18m00,t18m01,t18m02,t18m03,t18m04,t18m05,t18m06,t18m07,t18m08,t18m09,
t18m10,t18m11,t18m12,t18m13,t18m14,t18m15,t18m16,t18m17,t18m18,t18m19,t
18m20,t18m21,t18m22,t18m23);
[t16mm,t16ms,t16mt,t16mmn,t16msn] = meanstdti
(t16m00,t16m01,t16m02,t16m03,t16m04,t16m05,t16m06,t16m07,t16m08,t16m09,
t16m10,t16m11,t16m12,t16m13,t16m14,t16m15,t16m16,t16m17,t16m18,t16m19,t
16m20,t16m21,t16m22,t16m23);
[t14mm,t14ms,t14mt,t14mmn,t14msn] = meanstdti
(t14m00,t14m01,t14m02,t14m03,t14m04,t14m05,t14m06,t14m07,t14m08,t14m09,
t14m10,t14m11,t14m12,t14m13,t14m14,t14m15,t14m16,t14m17,t14m18,t14m19,t
14m20,t14m21,t14m22,t14m23);
[t12mm,t12ms,t12mt,t12mmn,t12msn] = meanstdti
(t12m00,t12m01,t12m02,t12m03,t12m04,t12m05,t12m06,t12m07,t12m08,t12m09,
t12m10,t12m11,t12m12,t12m13,t12m14,t12m15,t12m16,t12m17,t12m18,t12m19,t
12m20,t12m21,t12m22,t12m23);

%1 year data
[t18ym,t18ys,t18yt,t18ymn,t18ysn] = meanstdti
(t18y00,t18y01,t18y02,t18y03,t18y04,t18y05,t18y06,t18y07,t18y08,t18y09,
t18y10,t18y11,t18y12,t18y13,t18y14,t18y15,t18y16,t18y17,t18y18,t18y19,t
18y20,t18y21,t18y22,t18y23);
[t16ym,t16ys,t16yt,t16ymn,t16ysn] = meanstdti
(t16y00,t16y01,t16y02,t16y03,t16y04,t16y05,t16y06,t16y07,t16y08,t16y09,
t16y10,t16y11,t16y12,t16y13,t16y14,t16y15,t16y16,t16y17,t16y18,t16y19,t
16y20,t16y21,t16y22,t16y23);
[t14ym,t14ys,t14yt,t14ymn,t14ysn] = meanstdti
(t14y00,t14y01,t14y02,t14y03,t14y04,t14y05,t14y06,t14y07,t14y08,t14y09,
t14y10,t14y11,t14y12,t14y13,t14y14,t14y15,t14y16,t14y17,t14y18,t14y19,t
14y20,t14y21,t14y22,t14y23);
[t12ym,t12ys,t12yt,t12ymn,t12ysn] = meanstdti
(t12y00,t12y01,t12y02,t12y03,t12y04,t12y05,t12y06,t12y07,t12y08,t12y09,
t12y10,t12y11,t12y12,t12y13,t12y14,t12y15,t12y16,t12y17,t12y18,t12y19,t
12y20,t12y21,t12y22,t12y23);

%Calculating the mean of the of the mean and std. dev. vectors from
above.
%The vectors where the "NaN" values have been removed are used becuse
%errors occured when trying to take the mean of vectors containing
"NaN"
%These values will be used for the purpose of calculating the Rayleigh
and
%Weibull distributions for the data.
%Notation: (The trailing "n" has been moved to the front from the rear
to
%indicate that these variables are now single values rather than
vectors.)
nt18dm = mean(t18dmn);
nt18wm = mean(t18wmn);
nt18mm = mean(t18mmn);
nt18ym = mean(t18ymn);

nt18ds = mean(t18dsn);
nt18ws = mean(t18wsn);
nt18ms = mean(t18msn);
```

```
nt18ys = mean(t18ysn);

nt16dm = mean(t16dmn);
nt16wm = mean(t16wmn);
nt16mm = mean(t16mmn);
nt16ym = mean(t16ymn);

nt16ds = mean(t16dsn);
nt16ws = mean(t16wsn);
nt16ms = mean(t16msn);
nt16ys = mean(t16ysn);

nt14dm = mean(t14dmn);
nt14wm = mean(t14wmn);
nt14mm = mean(t14mmn);
nt14ym = mean(t14ymn);

nt14ds = mean(t14dsn);
nt14ws = mean(t14wsn);
nt14ms = mean(t14msn);
nt14ys = mean(t14ysn);

nt12dm = mean(t12dmn);
nt12wm = mean(t12wmn);
nt12mm = mean(t12mmn);
nt12ym = mean(t12ymn);

nt12ds = mean(t12dsn);
nt12ws = mean(t12wsn);
nt12ms = mean(t12msn);
nt12ys = mean(t12ysn);

%Calculating the rayleigh and weibull distributions for the date-wise
%parsed data by calling two custom functions I wrote called
"rayleighpdf"
%and "weibullpdf".
%Notation: (ray stands for rayleigh and weib stands for weibull, the
next
%two digits correspond to tower height, and the last character
corresponds
%to date-wise parsing) These are vectors now.
[ray18d] = rayleighpdf (nt18dm);
[ray18w] = rayleighpdf (nt18wm);
[ray18m] = rayleighpdf (nt18mm);
[ray18y] = rayleighpdf (nt18ym);

[ray16d] = rayleighpdf (nt16dm);
[ray16w] = rayleighpdf (nt16wm);
[ray16m] = rayleighpdf (nt16mm);
[ray16y] = rayleighpdf (nt16ym);

[ray14d] = rayleighpdf (nt14dm);
[ray14w] = rayleighpdf (nt14wm);
[ray14m] = rayleighpdf (nt14mm);
[ray14y] = rayleighpdf (nt14ym);
```

```
[ray12d] = rayleighpdf (nt12dm);
[ray12w] = rayleighpdf (nt12wm);
[ray12m] = rayleighpdf (nt12mm);
[ray12y] = rayleighpdf (nt12ym);

[weib18d] = weibullpdf (nt18dm, nt18ds);
[weib18w] = weibullpdf (nt18wm, nt18ws);
[weib18m] = weibullpdf (nt18mm, nt18ms);
[weib18y] = weibullpdf (nt18ym, nt18ys);

[weib16d] = weibullpdf (nt16dm, nt16ds);
[weib16w] = weibullpdf (nt16wm, nt16ws);
[weib16m] = weibullpdf (nt16mm, nt16ms);
[weib16y] = weibullpdf (nt16ym, nt16ys);

[weib14d] = weibullpdf (nt14dm, nt14ds);
[weib14w] = weibullpdf (nt14wm, nt14ws);
[weib14m] = weibullpdf (nt14mm, nt14ms);
[weib14y] = weibullpdf (nt14ym, nt14ys);

[weib12d] = weibullpdf (nt12dm, nt12ds);
[weib12w] = weibullpdf (nt12wm, nt12ws);
[weib12m] = weibullpdf (nt12mm, nt12ms);
[weib12y] = weibullpdf (nt12ym, nt12ys);

%Creating plots to visualize the data
%Plotting Rayleigh and Weibull distributions for all tower heights
together
%along with histograms for each tower height for a total of six
subplots.
%Each of these six subplots are created for each date-wise duration for
a
%total of 4 figures each containing these 6 subplots.
u = [0:0.1:25];
bins = [0:1:25];

f1 = figure(1);
set(f1,'units','normalized','position',[0 0.05 1 0.86]);
subplot(3,2,1);
plot(u, ray18d, u, ray16d, u, ray14d, u, ray12d);
xlabel('Wind Speed (mph)');
ylabel('Probability');
title('Rayleigh Probability Density Function (Yesterday)');
legend('80ft','60ft','40ft','20ft');
axis([0 25 0 0.2]);

subplot(3,2,2);
plot(u, weib18d, u, weib16d, u, weib14d, u, weib12d);
xlabel('Wind Speed (mph)');
ylabel('Probability');
title('Weibull Probability Density Function (Yesterday)');
legend('80ft','60ft','40ft','20ft');
axis([0 25 0 0.2]);
```

```
subplot(3,2,3);
n_elements80 = hist(tower18d(:,3), bins);
c_elements80 = numel(tower18d(:,3));
p_elements80 = n_elements80./c_elements80;
bar(bins, p_elements80);
xlabel('Wind Speed (mph)');
ylabel('Percent');
title('Histogram 80ft');
axis([0 25 0 0.2]);

subplot(3,2,4);
n_elements60 = hist(tower16d(:,3), bins);
c_elements60 = numel(tower16d(:,3));
p_elements60 = n_elements60./c_elements60;
bar(bins, p_elements60);
xlabel('Wind Speed (mph)');
ylabel('Percent');
title('Histogram 60ft');
axis([0 25 0 0.2]);

subplot(3,2,5);
n_elements40 = hist(tower14d(:,3), bins);
c_elements40 = numel(tower14d(:,3));
p_elements40 = n_elements40./c_elements40;
bar(bins, p_elements40);
xlabel('Wind Speed (mph)');
ylabel('Percent');
title('Histogram 40ft');
axis([0 25 0 0.2]);

subplot(3,2,6);
n_elements20 = hist(tower12d(:,3), bins);
c_elements20 = numel(tower12d(:,3));
p_elements20 = n_elements20./c_elements20;
bar(bins, p_elements20);
xlabel('Wind Speed (mph)');
ylabel('Percent');
title('Histogram 20ft');
axis([0 25 0 0.2]);


f2 = figure(2);
set(f2,'units','normalized','position',[0 0.05 1 0.86]);
subplot(3,2,1);
plot(u, ray18w, u, ray16w, u, ray14w, u, ray12w);
xlabel('Wind Speed (mph)');
ylabel('Probability');
title('Rayleigh Probability Density Function (Last 7 Days)');
legend('80ft','60ft','40ft','20ft');
axis([0 25 0 0.2]);

subplot(3,2,2);
plot(u, weib18w, u, weib16w, u, weib14w, u, weib12w);
xlabel('Wind Speed (mph)');
ylabel('Probability');
title('Weibull Probability Density Function (Last 7 Days)');
```

```
legend('80ft','60ft','40ft','20ft');
axis([0 25 0 0.2]);

subplot(3,2,3);
n_elements80 = hist(tower18w(:,3), bins);
c_elements80 = numel(tower18w(:,3));
p_elements80 = n_elements80./c_elements80;
bar(bins, p_elements80);
xlabel('Wind Speed (mph)');
ylabel('Percent');
title('Histogram 80ft');
axis([0 25 0 0.2]);

subplot(3,2,4);
n_elements60 = hist(tower16w(:,3), bins);
c_elements60 = numel(tower16w(:,3));
p_elements60 = n_elements60./c_elements60;
bar(bins, p_elements60);
xlabel('Wind Speed (mph)');
ylabel('Percent');
title('Histogram 60ft');
axis([0 25 0 0.2]);

subplot(3,2,5);
n_elements40 = hist(tower14w(:,3), bins);
c_elements40 = numel(tower14w(:,3));
p_elements40 = n_elements40./c_elements40;
bar(bins, p_elements40);
xlabel('Wind Speed (mph)');
ylabel('Percent');
title('Histogram 40ft');
axis([0 25 0 0.2]);

subplot(3,2,6);
n_elements20 = hist(tower12w(:,3), bins);
c_elements20 = numel(tower12w(:,3));
p_elements20 = n_elements20./c_elements20;
bar(bins, p_elements20);
xlabel('Wind Speed (mph)');
ylabel('Percent');
title('Histogram 20ft');
axis([0 25 0 0.2]);


f3 = figure(3);
set(f3,'units','normalized','position',[0 0.05 1 0.86]);
subplot(3,2,1);
plot(u, ray18m, u, ray16m, u, ray14m, u, ray12m);
xlabel('Wind Speed (mph)');
ylabel('Probability');
title('Rayleigh Probability Density Function (Last 30 Days)');
legend('80ft','60ft','40ft','20ft');
axis([0 25 0 0.2]);

subplot(3,2,2);
```

```
plot(u, weib18m, u, weib16m, u, weib14m, u, weib12m);
xlabel('Wind Speed (mph)');
ylabel('Probability');
title('Weibull Probability Density Function (Last 30 Days)');
legend('80ft','60ft','40ft','20ft');
axis([0 25 0 0.2]);

subplot(3,2,3);
n_elements80 = hist(tower18m(:,3), bins);
c_elements80 = numel(tower18m(:,3));
p_elements80 = n_elements80./c_elements80;
bar(bins, p_elements80);
xlabel('Wind Speed (mph)');
ylabel('Percent');
title('Histogram 80ft');
axis([0 25 0 0.2]);

subplot(3,2,4);
n_elements60 = hist(tower16m(:,3), bins);
c_elements60 = numel(tower16m(:,3));
p_elements60 = n_elements60./c_elements60;
bar(bins, p_elements60);
xlabel('Wind Speed (mph)');
ylabel('Percent');
title('Histogram 60ft');
axis([0 25 0 0.2]);

subplot(3,2,5);
n_elements40 = hist(tower14m(:,3), bins);
c_elements40 = numel(tower14m(:,3));
p_elements40 = n_elements40./c_elements40;
bar(bins, p_elements40);
xlabel('Wind Speed (mph)');
ylabel('Percent');
title('Histogram 40ft');
axis([0 25 0 0.2]);

subplot(3,2,6);
n_elements20 = hist(tower12m(:,3), bins);
c_elements20 = numel(tower12m(:,3));
p_elements20 = n_elements20./c_elements20;
bar(bins, p_elements20);
xlabel('Wind Speed (mph)');
ylabel('Percent');
title('Histogram 20ft');
axis([0 25 0 0.2]);

f4 = figure(4);
set(f4,'units','normalized','position',[0 0.05 1 0.86]);
subplot(3,2,1);
plot(u, ray18y, u, ray16y, u, ray14y, u, ray12y);
xlabel('Wind Speed (mph)');
ylabel('Probability');
title('Rayleigh Probability Density Function (Last 365 Days)');
legend('80ft','60ft','40ft','20ft');
axis([0 25 0 0.2]);
```

```
subplot(3,2,2);
plot(u, weib18y, u, weib16y, u, weib14y, u, weib12y);
xlabel('Wind Speed (mph)');
ylabel('Probability');
title('Weibull Probability Density Function (Last 365 Days)');
legend('80ft','60ft','40ft','20ft');
axis([0 25 0 0.2]);

subplot(3,2,3);
n_elements80 = hist(tower18y(:,3), bins);
c_elements80 = numel(tower18y(:,3));
p_elements80 = n_elements80./c_elements80;
bar(bins, p_elements80);
xlabel('Wind Speed (mph)');
ylabel('Percent');
title('Histogram 80ft');
axis([0 25 0 0.2]);

subplot(3,2,4);
n_elements60 = hist(tower16y(:,3), bins);
c_elements60 = numel(tower16y(:,3));
p_elements60 = n_elements60./c_elements60;
bar(bins, p_elements60);
xlabel('Wind Speed (mph)');
ylabel('Percent');
title('Histogram 60ft');
axis([0 25 0 0.2]);

subplot(3,2,5);
n_elements40 = hist(tower14y(:,3), bins);
c_elements40 = numel(tower14y(:,3));
p_elements40 = n_elements40./c_elements40;
bar(bins, p_elements40);
xlabel('Wind Speed (mph)');
ylabel('Percent');
title('Histogram 40ft');
axis([0 25 0 0.2]);

subplot(3,2,6);
n_elements20 = hist(tower12y(:,3), bins);
c_elements20 = numel(tower12y(:,3));
p_elements20 = n_elements20./c_elements20;
bar(bins, p_elements20);
xlabel('Wind Speed (mph)');
ylabel('Percent');
title('Histogram 20ft');
axis([0 25 0 0.2]);

%Preparing data to be integrated for the purpose of finding the average
%wind speed
hour = [0:1:23];

%Computing which indices of the input vectors are "NaN"
i18dm = find(~isnan(t18dm));
```

```
i16dm = find(~isnan(t16dm));
i14dm = find(~isnan(t14dm));
i12dm = find(~isnan(t12dm));

i18wm = find(~isnan(t18wm));
i16wm = find(~isnan(t16wm));
i14wm = find(~isnan(t14wm));
i12wm = find(~isnan(t12wm));

i18mm = find(~isnan(t18mm));
i16mm = find(~isnan(t16mm));
i14mm = find(~isnan(t14mm));
i12mm = find(~isnan(t12mm));

i18ym = find(~isnan(t18ym));
i16ym = find(~isnan(t16ym));
i14ym = find(~isnan(t14ym));
i12ym = find(~isnan(t12ym));

%Modifying the input vectors to only include the indices which aren't
"NaN"
at18dm = t18dm(i18dm);
at16dm = t16dm(i16dm);
at14dm = t14dm(i14dm);
at12dm = t12dm(i12dm);

at18wm = t18wm(i18wm);
at16wm = t16wm(i16wm);
at14wm = t14wm(i14wm);
at12wm = t12wm(i12wm);

at18mm = t18mm(i18mm);
at16mm = t16mm(i16mm);
at14mm = t14mm(i14mm);
at12mm = t12mm(i12mm);

at18ym = t18ym(i18ym);
at16ym = t16ym(i16ym);
at14ym = t14ym(i14ym);
at12ym = t12ym(i12ym);

%Modifying the "hour" vector to include the same indicies as the input
%vectors
hour18dm = hour(i18dm);
hour16dm = hour(i16dm);
hour14dm = hour(i14dm);
hour12dm = hour(i12dm);

hour18wm = hour(i18wm);
hour16wm = hour(i16wm);
hour14wm = hour(i14wm);
hour12wm = hour(i12wm);

hour18mm = hour(i18mm);
hour16mm = hour(i16mm);
```

```
hour14mm = hour(i14mm);
hour12mm = hour(i12mm);

hour18ym = hour(i18ym);
hour16ym = hour(i16ym);
hour14ym = hour(i14ym);
hour12ym = hour(i12ym);

%Calculating the area under each of the curves using a trapezoidal
%approximation
area18dm = trapz(hour18dm,at18dm);
area16dm = trapz(hour16dm,at16dm);
area14dm = trapz(hour14dm,at14dm);
area12dm = trapz(hour12dm,at12dm);

area18wm = trapz(hour18wm,at18wm);
area16wm = trapz(hour16wm,at16wm);
area14wm = trapz(hour14wm,at14wm);
area12wm = trapz(hour12wm,at12wm);

area18mm = trapz(hour18mm,at18mm);
area16mm = trapz(hour16mm,at16mm);
area14mm = trapz(hour14mm,at14mm);
area12mm = trapz(hour12mm,at12mm);

area18ym = trapz(hour18ym,at18ym);
area16ym = trapz(hour16ym,at16ym);
area14ym = trapz(hour14ym,at14ym);
area12ym = trapz(hour12ym,at12ym);

%Finally calculating the mean wind speed by dividing the area under the
%curve by the number of entries and then creating a vector populated
with
%the mean wind speed value
wind18dm = area18dm/numel(at18dm);
wind16dm = area16dm/numel(at16dm);
wind14dm = area14dm/numel(at14dm);
wind12dm = area12dm/numel(at12dm);
wind18dm(1,1:numel(at18dm)) = wind18dm;
wind16dm(1,1:numel(at16dm)) = wind16dm;
wind14dm(1,1:numel(at14dm)) = wind14dm;
wind12dm(1,1:numel(at12dm)) = wind12dm;

wind18wm = area18wm/numel(at18wm);
wind16wm = area16wm/numel(at16wm);
wind14wm = area14wm/numel(at14wm);
wind12wm = area12wm/numel(at12wm);
wind18wm(1,1:numel(at18wm)) = wind18wm;
wind16wm(1,1:numel(at16wm)) = wind16wm;
wind14wm(1,1:numel(at14wm)) = wind14wm;
wind12wm(1,1:numel(at12wm)) = wind12wm;

wind18mm = area18mm/numel(at18mm);
wind16mm = area16mm/numel(at16mm);
wind14mm = area14mm/numel(at14mm);
```

```
wind12mm = area12mm/numel(at12mm);
wind18mm(1,1:numel(at18mm)) = wind18mm;
wind16mm(1,1:numel(at16mm)) = wind16mm;
wind14mm(1,1:numel(at14mm)) = wind14mm;
wind12mm(1,1:numel(at12mm)) = wind12mm;

wind18ym = area18ym/numel(at18ym);
wind16ym = area16ym/numel(at16ym);
wind14ym = area14ym/numel(at14ym);
wind12ym = area12ym/numel(at12ym);
wind18ym(1,1:numel(at18ym)) = wind18ym;
wind16ym(1,1:numel(at16ym)) = wind16ym;
wind14ym(1,1:numel(at14ym)) = wind14ym;
wind12ym(1,1:numel(at12ym)) = wind12ym;

%Plotting hourly mean wind speed and turbulence intensity for each
%date-wise duration for a total of 4 figures each containing 2
subplots.
f5 = figure(5);
set(f5,'units','normalized','position',[0 0.05 1 0.86]);
subplot(1,2,1);
plot(hour, t18dm, hour, t16dm, hour, t14dm, hour, t12dm, hour18dm,
wind18dm, '--b', hour16dm, wind16dm, '--k', hour14dm, wind14dm, '--r',
hour12dm, wind12dm, '--c');
xlabel('Hour');
ylabel('Mean Wind Speed (mph)');
title('Hourly Mean Wind Speeds (Yesterday)');
legend('80ft','60ft','40ft','20ft','Mean 80ft','Mean 60ft','Mean
40ft','Mean 20ft','Location','NE');
grid on;
axis ([0 23 0 16]);

subplot(1,2,2);
plot(hour, t18dt, hour, t16dt, hour, t14dt, hour, t12dt);
xlabel('Hour');
ylabel('Turbulence Intensity');
title('Hourly Turbulence Intensity (Yesterday)');
legend('80ft','60ft','40ft','20ft','Location','NE');

f6 = figure(6);
set(f6,'units','normalized','position',[0 0.05 1 0.86]);
subplot(1,2,1);
plot(hour, t18wm, hour, t16wm, hour, t14wm, hour, t12wm, hour18wm,
wind18wm, '--b', hour16wm, wind16wm, '--k', hour14wm, wind14wm, '--r',
hour12wm, wind12wm, '--c');
xlabel('Hour');
ylabel('Mean Wind Speed (mph)');
title('Hourly Mean Wind Speeds (Last 7 Days)');
legend('80ft','60ft','40ft','20ft','Mean 80ft','Mean 60ft','Mean
40ft','Mean 20ft','Location','NE');
grid on;
axis ([0 23 0 16]);

subplot(1,2,2);
plot(hour, t18wt, hour, t16wt, hour, t14wt, hour, t12wt);
xlabel('Hour');
```

```
ylabel('Turbulence Intensity');
title('Hourly Turbulence Intensity (Last 7 Days)');
legend('80ft','60ft','40ft','20ft','Location','NE');

f7 = figure(7);
set(f7,'units','normalized','position',[0 0.05 1 0.86]);
subplot(1,2,1);
plot(hour, t18mm, hour, t16mm, hour, t14mm, hour, t12mm, hour18mm,
wind18mm, '--b', hour16mm, wind16mm, '--k', hour14mm, wind14mm, '--r',
hour12mm, wind12mm, '--c');
xlabel('Hour');
ylabel('Mean Wind Speed (mph)');
title('Hourly Mean Wind Speeds (Last 30 Days)');
legend('80ft','60ft','40ft','20ft','Mean 80ft','Mean 60ft','Mean
40ft','Mean 20ft','Location','NE');
grid on;
axis ([0 23 0 16]);

subplot(1,2,2);
plot(hour, t18mt, hour, t16mt, hour, t14mt, hour, t12mt);
xlabel('Hour');
ylabel('Turbulence Intensity');
title('Hourly Turbulence Intensity (Last 30 Days)');
legend('80ft','60ft','40ft','20ft','Location','NE');

f8 = figure(8);
set(f8,'units','normalized','position',[0 0.05 1 0.86]);
subplot(1,2,1);
plot(hour, t18ym, hour, t16ym, hour, t14ym, hour, t12ym, hour18ym,
wind18ym, '--b', hour16ym, wind16ym, '--k', hour14ym, wind14ym, '--r',
hour12ym, wind12ym, '--c');
xlabel('Hour');
ylabel('Mean Wind Speed (mph)');
title('Hourly Mean Wind Speeds (Last 365 Days)');
legend('80ft','60ft','40ft','20ft','Mean 80ft','Mean 60ft','Mean
40ft','Mean 20ft','Location','NE');
grid on;
axis ([0 23 0 16]);

subplot(1,2,2);
plot(hour, t18yt, hour, t16yt, hour, t14yt, hour, t12yt);
xlabel('Hour');
ylabel('Turbulence Intensity');
title('Hourly Turbulence Intensity (Last 365 Days)');
legend('80ft','60ft','40ft','20ft','Location','NE');

%Saving each figure as a jpeg image in the location specified in
purple.
saveas(f1,'T:/Cal_Poly_Wind_Data/PDFs (Yesterday).jpg');
saveas(f2,'T:/Cal_Poly_Wind_Data/PDFs (Last 7 Days).jpg');
saveas(f3,'T:/Cal_Poly_Wind_Data/PDFs (Last 30 Days).jpg');
saveas(f4,'T:/Cal_Poly_Wind_Data/PDFs (Last 365 Days).jpg');
saveas(f5,'T:/Cal_Poly_Wind_Data/Mean and TI (Yesterday).jpg');
saveas(f6,'T:/Cal_Poly_Wind_Data/Mean and TI (Last 7 Days).jpg');
saveas(f7,'T:/Cal_Poly_Wind_Data/Mean and TI (Last 30 Days).jpg');
saveas(f8,'T:/Cal_Poly_Wind_Data/Mean and TI (Last 365 Days).jpg');
```

```
function
[h00,h01,h02,h03,h04,h05,h06,h07,h08,h09,h10,h11,h12,h13,h14,h15,h16,h1
7,h18,h19,h20,h21,h22,h23] = hourfilter (mon00)
%This function will filter the data by hour

hmon00 = sortrows(mon00);

nmax = numel(mon00(:,1));
n00 = 1;
n01 = 1;
n02 = 1;
n03 = 1;
n04 = 1;
n05 = 1;
n06 = 1;
n07 = 1;
n08 = 1;
n09 = 1;
n10 = 1;
n11 = 1;
n12 = 1;
n13 = 1;
n14 = 1;
n15 = 1;
n16 = 1;
n17 = 1;
n18 = 1;
n19 = 1;
n20 = 1;
n21 = 1;
n22 = 1;
n23 = 1;

while (nmax >= n00) && (hmon00(n00,1) < 1/24);
    n00 = n00+1;
end
n00 = n00-1;

while (nmax >= n01) && (hmon00(n01,1) < 2/24);
    n01 = n01+1;
end
n01 = n01-1;

while (nmax >= n02) && (hmon00(n02,1) < 3/24);
    n02 = n02+1;
end
n02 = n02-1;

while (nmax >= n03) && (hmon00(n03,1) < 4/24);
    n03 = n03+1;
end
n03 = n03-1;

while (nmax >= n04) && (hmon00(n04,1) < 5/24);
    n04 = n04+1;
```

```
end
n04 = n04-1;

while (nmax >= n05) && (hmon00(n05,1) < 6/24);
    n05 = n05+1;
end
n05 = n05-1;

while (nmax >= n06) && (hmon00(n06,1) < 7/24);
    n06 = n06+1;
end
n06 = n06-1;

while (nmax >= n07) && (hmon00(n07,1) < 8/24);
    n07 = n07+1;
end
n07 = n07-1;

while (nmax >= n08) && (hmon00(n08,1) < 9/24);
    n08 = n08+1;
end
n08 = n08-1;

while (nmax >= n09) && (hmon00(n09,1) < 10/24);
    n09 = n09+1;
end
n09 = n09-1;

while (nmax >= n10) && (hmon00(n10,1) < 11/24);
    n10 = n10+1;
end
n10 = n10-1;

while (nmax >= n11) && (hmon00(n11,1) < 12/24);
    n11 = n11+1;
end
n11 = n11-1;

while (nmax >= n12) && (hmon00(n12,1) < 13/24);
    n12 = n12+1;
end
n12 = n12-1;

while (nmax >= n13) && (hmon00(n13,1) < 14/24);
    n13 = n13+1;
end
n13 = n13-1;

while (nmax >= n14) && (hmon00(n14,1) < 15/24);
    n14 = n14+1;
end
n14 = n14-1;

while (nmax >= n15) && (hmon00(n15,1) < 16/24);
    n15 = n15+1;
```

```
end
n15 = n15-1;

while (nmax >= n16) && (hmon00(n16,1) < 17/24);
    n16 = n16+1;
end
n16 = n16-1;

while (nmax >= n17) && (hmon00(n17,1) < 18/24);
    n17 = n17+1;
end
n17 = n17-1;

while (nmax >= n18) && (hmon00(n18,1) < 19/24);
    n18 = n18+1;
end
n18 = n18-1;

while (nmax >= n19) && (hmon00(n19,1) < 20/24);
    n19 = n19+1;
end
n19 = n19-1;

while (nmax >= n20) && (hmon00(n20,1) < 21/24);
    n20 = n20+1;
end
n20 = n20-1;

while (nmax >= n21) && (hmon00(n21,1) < 22/24);
    n21 = n21+1;
end
n21 = n21-1;

while (nmax >= n22) && (hmon00(n22,1) < 23/24);
    n22 = n22+1;
end
n22 = n22-1;

while (nmax >= n23) && (hmon00(n23,1) < 1);
    n23 = n23+1;
end
n23 = n23-1;

h00 = hmon00(1:n00,2);
h01 = hmon00((n00+1):n01,2);
h02 = hmon00((n01+1):n02,2);
h03 = hmon00((n02+1):n03,2);
h04 = hmon00((n03+1):n04,2);
h05 = hmon00((n04+1):n05,2);
h06 = hmon00((n05+1):n06,2);
h07 = hmon00((n06+1):n07,2);
h08 = hmon00((n07+1):n08,2);
h09 = hmon00((n08+1):n09,2);
h10 = hmon00((n09+1):n10,2);
h11 = hmon00((n10+1):n11,2);
```

```
h12 = hmon00((n11+1):n12,2);
h13 = hmon00((n12+1):n13,2);
h14 = hmon00((n13+1):n14,2);
h15 = hmon00((n14+1):n15,2);
h16 = hmon00((n15+1):n16,2);
h17 = hmon00((n16+1):n17,2);
h18 = hmon00((n17+1):n18,2);
h19 = hmon00((n18+1):n19,2);
h20 = hmon00((n19+1):n20,2);
h21 = hmon00((n20+1):n21,2);
h22 = hmon00((n21+1):n22,2);
h23 = hmon00((n22+1):n23,2);
```

```
function [txm,txs,txt,txmn,txsn] = meanstdti
(tx00,tx01,tx02,tx03,tx04,tx05,tx06,tx07,tx08,tx09,tx10,tx11,tx12,tx13,
tx14,tx15,tx16,tx17,tx18,tx19,tx20,tx21,tx22,tx23)
%This function will calculate the mean, standard deviation, and
turbulence
%intensity for the monthly data

%Calculating the mean wind speed
tx00m = mean(tx00(:,1));
tx01m = mean(tx01(:,1));
tx02m = mean(tx02(:,1));
tx03m = mean(tx03(:,1));
tx04m = mean(tx04(:,1));
tx05m = mean(tx05(:,1));
tx06m = mean(tx06(:,1));
tx07m = mean(tx07(:,1));
tx08m = mean(tx08(:,1));
tx09m = mean(tx09(:,1));
tx10m = mean(tx10(:,1));
tx11m = mean(tx11(:,1));
tx12m = mean(tx12(:,1));
tx13m = mean(tx13(:,1));
tx14m = mean(tx14(:,1));
tx15m = mean(tx15(:,1));
tx16m = mean(tx16(:,1));
tx17m = mean(tx17(:,1));
tx18m = mean(tx18(:,1));
tx19m = mean(tx19(:,1));
tx20m = mean(tx20(:,1));
tx21m = mean(tx21(:,1));
tx22m = mean(tx22(:,1));
tx23m = mean(tx23(:,1));
txm = [tx00m tx01m tx02m tx03m tx04m tx05m tx06m tx07m tx08m tx09m
tx10m tx11m tx12m tx13m tx14m tx15m tx16m tx17m tx18m tx19m tx20m tx21m
tx22m tx23m];
txmn = txm(~isnan(txm));

%Calculating the standard deviation
tx00s = std(tx00(:,1));
tx01s = std(tx01(:,1));
tx02s = std(tx02(:,1));
tx03s = std(tx03(:,1));
tx04s = std(tx04(:,1));
tx05s = std(tx05(:,1));
tx06s = std(tx06(:,1));
tx07s = std(tx07(:,1));
tx08s = std(tx08(:,1));
tx09s = std(tx09(:,1));
tx10s = std(tx10(:,1));
tx11s = std(tx11(:,1));
tx12s = std(tx12(:,1));
tx13s = std(tx13(:,1));
tx14s = std(tx14(:,1));
tx15s = std(tx15(:,1));
tx16s = std(tx16(:,1));
tx17s = std(tx17(:,1));
```

```
tx18s = std(tx18(:,1));
tx19s = std(tx19(:,1));
tx20s = std(tx20(:,1));
tx21s = std(tx21(:,1));
tx22s = std(tx22(:,1));
tx23s = std(tx23(:,1));
txs = [tx00s tx01s tx02s tx03s tx04s tx05s tx06s tx07s tx08s tx09s
tx10s tx11s tx12s tx13s tx14s tx15s tx16s tx17s tx18s tx19s tx20s tx21s
tx22s tx23s];
txsn = txs(~isnan(txs));

%Caluclating the turbulence intensity
txt = txs./txm;
```

```
function [raypdf] = rayleighpdf (meanspeed);
%This function computes the Rayleigh probability density function based
on
%the mean wind speed

u = [0:0.1:25];

raypdf = (pi/2).*(u./(meanspeed^2)).*exp(-(pi/4).*(u./meanspeed).^2);


function [weibpdf] = weibullpdf (meanspeed, stdev)
%This function computes the Weibull pdf based on mean wind speed and
%standard deviation

u = [0:0.1:25];

k = (stdev/meanspeed)^-1.086;

c = meanspeed*(0.568+(0.433/k))^(-1/k);

weibpdf = (k/c).*(u./c).^(k-1).*exp(-(u./c).^k);


function [inmatrix] = removedata2(inmatrix,limit)
%This function will remove data that is beyond a specified limit from a
%given matrix

z = numel(inmatrix(:,1));

for column = [4 5 6]
    for i = 1:z
        if(inmatrix(i,column)>=limit)
            inmatrix(i,column)=NaN;
        end
    end
end

inmatrix(any(isnan(inmatrix),2),:)=[];
```

Nomenclature

∇: gradient operator

∇^2: Laplacian operator

α: 1.) wind shear exponent, when unknown it is customarily set to α=1/7
2.) empirical constant used in the *K*-ω turbulence model, typically equal to 5/9

β: empirical constant used in the *K*-ω turbulence model, typically equal to 3/40

β^*: empirical constant used in the *K*-ω turbulence model, typically equal to 9/100

ε: turbulent dissipation rate

κ: Von-Karman constant which is equal to 0.41

μ: fluid viscosity

v_t: 1.) turbulent viscosity, defined in Equation 16
2.) ratio of K to ω in the *K*-ω turbulence model

π: mathematical constant approximately equal to 3.14159

ρ: fluid density

σ: 1.) empirical constant used in the *K*-ω turbulence model, typically equal to 1/2
2.) measured standard deviation of wind speed used for a Weibull distribution

σ^*: empirical constant used in the *K*-ω turbulence model, typically equal to 1/2

σ_ε: empirical constant used in the *K*-ε turbulence model, typically equal to 1.3

σ_K: empirical constant used in the *K*-ε turbulence model, typically equal to 1.0

τ_{ij}: Reynold's stress tensor, each component is defined in Equation 12

τ_w: shear stress at the surface

ϕ: variable used in an equation to represent that the equation can be applied to many different variables without change to the equation

$\bar{\phi}$: average component of a variable with respect to turbulence

ϕ': fluctuating component of a variable with respect to turbulence

ω: specific dissipation

c: 1.) speed of sound
2.) Weibull distribution parameter, defined in Equation 24

C_1: empirical constant used in the K-ε turbulence model, typically equal to 1.44

C_2: empirical constant used in the K-ε turbulence model, typically equal to 1.92

C_μ: empirical constant used in the K-ε turbulence model, typically equal to 0.09

g: acceleration due to gravity, 9.81m/s^2 or 32.17ft/s^2

I: turbulence intensity

k: Weibull distribution parameter, defined in Equation 23

K: turbulent kinetic energy

L: characteristic length based on geometry and flow conditions

M: Mach number, defined in Equation 5

$p(U)$: probability of a wind speed U in the context of a Rayleigh or Weibull distribution

P: pressure

t: time

T: period of integration

u: 1.) magnitude of the velocity at height z
2.) the x-component of velocity in a Cartesian coordinate system

u_o: magnitude of the reference velocity measured at reference height z_o

u_*: a parameter called friction velocity, defined in Equation 2

U: wind speed with a probability $p(U)$ in the context of a Rayleigh or Weibull distribution

$\bar{U}$: measured mean wind speed used to compute a Rayleigh or Weibull distribution

v: 1.) velocity vector
2.) y-component of velocity in a Cartesian coordinate system

w: z-component of velocity in a Cartesian coordinate system

z: height from the ground surface

z_o: 1.) reference height from the ground used to determine power law velocity profiles
2.) roughness length used to determine log law velocity profiles

Glossary

anemometer:	a device designed to measure the speed and sometimes direction of the wind
ASTER-GDEM:	a collaborative project between NASA and METI to collect global DEM data via satellite
CFD:	acronym for Computational Fluid Dynamics
DEM:	acronym for Digital Elevation Model
diurnal:	a difference in behavior between day and night
FLUENT:	computational fluid dynamics modeling software created by ANSYS
GDEM:	acronym for Global Digital Elevation Model
GSM:	acronym for Groupe Spécial Mobile which is a standard developed by the European Telecommunications Standards Institute for 2G cellular networks
LEED:	acronym for Leadership in Energy and Environmental Design
LIDAR:	acronym for LIght Detection And Ranging which uses pulses of light to determine distance to or other properties of a target
MATLAB:	a computer programming language and environment
mesh:	the spatial discretization of geometry for use in CFD simulations
MET:	abbreviation for the word meteorological in the context of meteorological tower
METI:	acronym for the Japanese Ministry of Economy, Trade, and Industry which is involved in the ASTER-GDEM project
NASA:	acronym for the National Aeronautics and Space Administration which is involved in the ASTER-GDEM project

NIST:	acronym for National Institute of Standards and Technology, an agency of the U.S. department of Commerce which develops standards for technology
PDE:	acronym for Partial Differential Equation
point cloud:	a large group of points individually defined by x, y, and z coordinates
RAMS:	acronym for Regional Atmospheric Modeling Simulation which is a mesoscale numerical weather prediction system developed by Colorado State University
RANS:	acronym for Reynolds Averaged Navier-Stokes, these equations represent the time averaged form of the Navier-Stokes equations
Rayleigh distribution:	wind speed probability distribution based on a measured mean wind speed
SODAR:	acronym for SOnic Detection And Ranging which measures the scattering of sound waves
topography:	the relief features or surface configuration of an area
utility scale:	an adjective describing a turbine with a rated power above 2MW
Weibull distribution:	wind speed probability distribution based on a measured mean wind speed and standard deviation
wind resource:	details about the wind such as speed and direction
wind rose:	for a given locality a graphical means of showing the strength and frequency of the wind from various directions
WRF:	acronym for Weather Research Forecasting which is a mesoscale numerical weather prediction system developed by a group of institutions including NCAR, NCEP and others

References

Alinot, C. & Masson, C. (2005). k-ε model for the atmospheric boundary layer under various thermal stratifications. *Journal of Solar Engineering, 127,* 438-443.

ANSYS (2006). Standard, RNG, and Realizable K-ε Models Theory. *Fluent 6.3 User's Guide,* Section 12.4.

ASTER-GDEM (2009). *NASA and METI.* <http://www.gdem.aster.ersdac.or.jp/> Downloaded: June 2010.

Bailey, B.H. et al. (1997). *Wind Resource Assessment Handbook.* Albany, NY: AWS Scientific

Bechrakis, D.A., & Deane, J.P., & McKeogh, E.J. (2004). Wind resource assessment of an area using short term data correlated to a long term data set. *Solar Energy, 76,* 725-732.

Bowen, A.J. (2003). Modelling of strong wind flows over complex terrain at small geometric scales. *Journal of Wind Engineering and Industrial Aerodynamics, 91,* 1859-1871.

Delwart, A. & Hölzer, M. (2010). Validity check of numerical site calibration according to IEC 61400-12-1 ed. 1 and MEASNET PPMP v4. *European Wind Energy Conference & Exhibition 2010.*

Eidsvik, K.J. (2005). A system for wind power estimation in mountainous terrain. Prediction of Askervein hill data. *Wind Energy, 8,* 237-249.

Elliot, D.K. (2010). Assistant Director Energy, Utilities, and Sustainability. *Facility Services California Polytechnic State University.* Summary of Excel workbook: Bank 1 FY0809 meter 1003692743.xls.

Forthofer, J.M. (2007). Modeling wind in complex terrain for use in fire spread prediction. *Colorado State University: Department of Forest Rangeland and Watershed Stewardship,* Thesis.

Grant, L.M. & Mason, P.J. (1990). Observations of boundary-layer structure over complex terrain. *Quarterly Journal of the Royal Meteorological Society, 116,* 159-186.

Griffiths, A.D. & Middleton, J.H. (2010). Simulations of separated flow over two-dimensional hills. *Journal of Wind Engineering and Industrial Aerodynamics, 98,* 155-160.

Guth, Peter (2010). MICRODEM. *United States Naval Academy.* <http://www.usna.edu/Users/oceano/pguth/website/microdem/microdem.htm> Downloaded: June 2010

IEC: 61400-12-1 (2005). Wind turbines – Power performance measurements of electricity producing wind turbines. *International Electrotechnical Commission, 1st Edition.* Geneva, Switzerland

Kim, Y.K. et al. (2008). Modeling the response of multi-scale winds in a mountainous coastal region to SST changes over the east sea in Korea. *American Meteorological Society 18th Symposium on Boundary Layers and Turbulence.*

Launder, B.E. & Spaulding, D.B. (1974). The numerical computation of turbulent flows. *Computer Methods in Applied Mechanics and Engineering, 3,* 269-289.

Li, Lei et al. (2010). Study on the micro-scale simulation of wind field over complex terrain by RAMS/FLUENT modeling system. *The Fifth International Symposium on Computational Wind Engineering.*

Manwell, J.F. et al. (2002). *Wind Energy Explained Theory, Design and Application.* Chichester, England: John Wiley & Sons Inc.

ME 554 (2011). Computational Heat Transfer. *Mechanical Engineering Department, California Polytechnic State University.* Professor Shollenberger.

Palma, J.M.L.M. et al. (2008). Linear and nonlinear models in wind resource assessment and wind turbine micro-siting in complex terrain. *Journal of Wind Engineering and Industrial Aerodynamics, 96,* 2308-2326.

Patel, V.C. et al. (2000). Numerical simulation of wind flow over hilly terrain. *Journal of Wind Engineering and Industrial Aerodynamics, 87,* 45-60.

Politis, E.S. & Chaviaropoulos, P.K. (2008). Micrositing and classification of wind turbines in complex terrain. *European Wind Energy Conference, CS4: Site Assessment.*

Prospathopoulos, J.M. et al. (2010). Simulation of wind farms in flat and complex terrain using CFD. *Torque 2010, The Science of Making Torque from Wind,* 359-370.

Ray, M.L. et al. (2006). Analysis of wind shear models and trends in different terrains. *American Wind Energy Association: Windpower 2005 Conference*

Russell, Alan. (2009). Computational fluid dynamics modeling of atmospheric flow applied to wind energy research. *Boise State University: Mechanical Engineering Department,* Thesis.

Taylor P.A. & Teunissen H.W. (1985). The Askervein Hill Project: Report on the Sept./Oct. 1983, Main Field Experiment. *Atmospheric Environment Service.*

White, F.M. (2006). *Viscous Fluid Flow.* New York: McGraw Hill

Wilcox, D.C. (1988). *Turbulence Modeling for CFD.* La Canada, CA: DCW Industries.

WindRose PRO (2011). *Enviroware.* <http://www.enviroware.com/windrose.htm> Downloaded: June 2010

Wood, Nigel (2000). Wind flow over complex terrain: A historical perspective and the prospect for large-eddy modelling. *Boundary Layer Meteorology, 96,* 11-32.